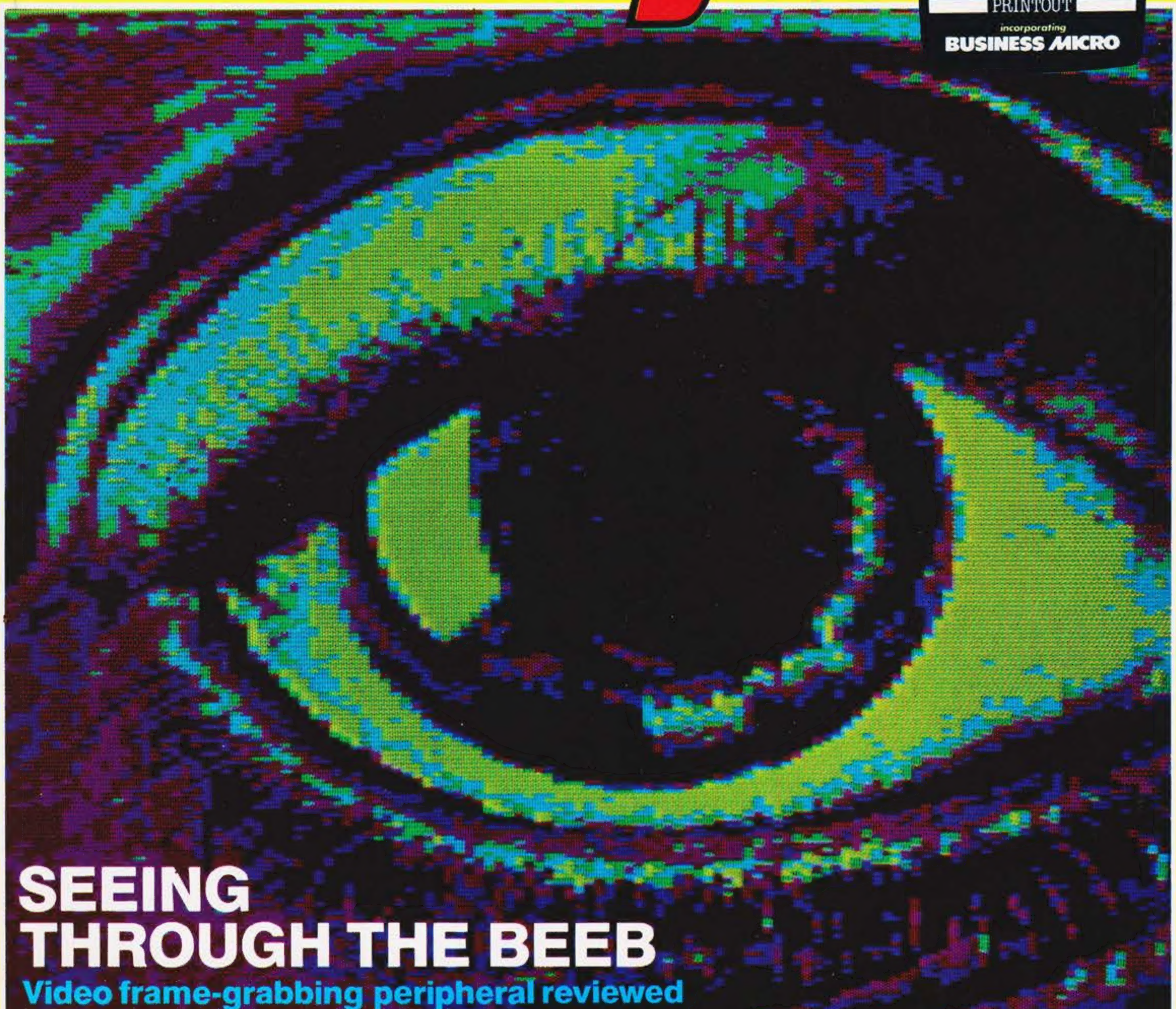
An Argus Specialist Publication

Computing today

JUNE 1984
90p

INCORPORATING
MICRO COMPUTER PRINTOUT
incorporating
BUSINESS MICRO



SEEING THROUGH THE BEEB

Video frame-grabbing peripheral reviewed

- A Feast of Fantasy: Adventures on the BBC Micro, Commodore 64, Atari, Spectrum and Dragon
- The hard facts about Winchester discs
- For the first time anywhere, our CP/M Directory guides you through the world's largest software selection

Plus

Our usual features and software for the home computer user

Little Brothers should be seen but not heard.

A maxim which eloquently describes the Brother HR-5.

Less than a foot across, it's nonetheless loaded with features.

But there's one thing the HR-5 won't give you. Earache.

For the annoying 'clickety clack' many printers produce is mercifully absent from the HR-5.

Quietly efficient, it delivers high definition dot matrix text over 80 columns at 30 c.p.s.

The HR-5 also has something of an artistic bent.

Being capable of producing uni-directional graph and chart images together with bi-directional text.

It will also hone down characters into a condensed face, or extend them for added emphasis.

Incorporating either a Centronics parallel or RS-232C interface, the HR-5 is compatible with most home computers and popular software.

Perfectly portable, the battery or mains operated HR-5 weighs less than 4lbs, and has a starting price of only £179.95 (inc.VAT).

Which is really something to shout about.

PLEASE SEND ME MORE DETAILS OF THE REMARKABLE BROTHER HR-5 PRINTER. CT6/84

NAME____________________

ADDRESS____________________

____

____

__________ TEL.NO.__________

brother®

DEPT P, BROTHER OFFICE EQUIPMENT DIVISION, JONES + BROTHER, SHEPLEY STREET, GUIDE BRIDGE, AUDENSHAW, MANCHESTER M34 5JD
TEL:061 330 6531 (10 LINES) 061 330 0111 (6 LINES) 061 330 3036 (4 LINES). TELEX: 669092. BROTHER INDUSTRIES LTD., NAGOYA, JAPAN

Editor:
Peter Green

Editorial Assistant:
Jamie Clary

Advertisement Manager:
Jeremy Ornellas

Advertisement Copy Control:
Sue Couchman,
Ann McDermott

Managing Editor:
Ron Harris BSc

Chief Executive:
T.J. Connell

Origination by Design International.

ABC Member of the Audit Bureau of Circulation

ISSN 0142-7210

Computing Today is normally published on the second Friday in the month preceding cover date. Distributed by: Argus Press Sales & Distribution Ltd, 12-18 Paul Street, London EC2A 4JS. 01-247 8233. Printed by: Alabaster Passmore & Sons Ltd, Maidstone, Kent.

Subscription rates: UK £13.30 including postage. Airmail and other rates upon application to Computing Today Subscriptions Department, Infonet Ltd, Times House, 179 The Marlowes, Hemel Hempstead, Herts HP1 1BB, England.

Computing Today is constantly on the look-out for well written articles and programs. If you think that your efforts meet our standards, please feel free to submit your work to us for consideration.

All material should be typed. Any programs submitted must be listed (cassette tapes and discs will not be accepted) and should be accompanied by sufficient documentation to enable their implementation. Please enclose an SAE if you want your manuscript returned, all submissions will be acknowledged. Any published work will be paid for.

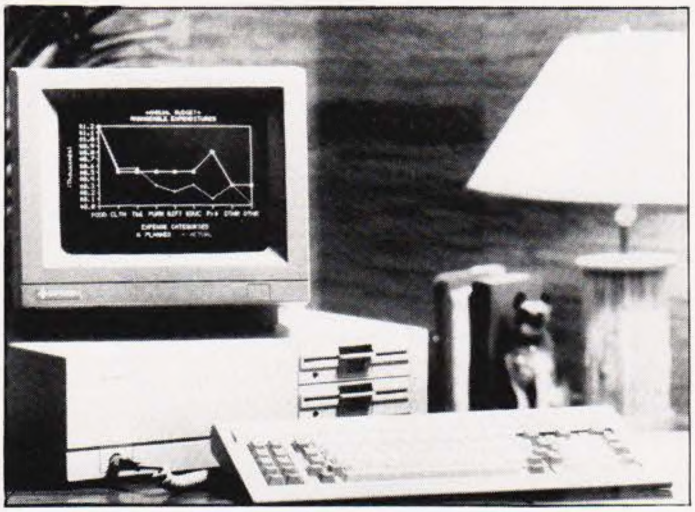
All work for consideration should be sent to the Editor at our Golden Square address.

CONTENTS

VOL 6 NO 4 JUNE 1984

EDITORIAL & ADVERTISEMENT OFFICE
No. 1, Golden Square, London W1R 3AB.
Telephone 01-437 0626. Telex 8811896.

NEWS



SPERRY INTERESTING

The Sperry Personal Computer, launched this month in the UK, is designed to be hardware and software compatible with the IBM PC and PC/XT but offers both functional and price advantages. As a stand-alone desktop 'PC', the Sperry Personal Computer takes advantage of IBM's 'open architecture' concept and taps the vast array of application programs utilising the highly-popular MS-DOS operating system.

In addition, the Sperry Personal Computer, with 128K bytes of resident memory, operates at 50% faster speeds than the IBM PC and XT personal computers. Sperry's computer also features the industry's most complete colour graphics with twice the resolution of the IBM models; an enhanced but compatible keyboard; a built-in, automatic time-of-day clock; and built-in asynchronous communication on all models.

Not only is the Sperry Personal Computer more powerful and price competitive as a stand-alone, it's also a single window to all the information that is produced by Sperry or IBM mainframes. In addition to communicating with other MS-DOS driven PC's using asynchronous communications, the Sperry Personal Computer can be used as a terminal to both Sperry and IBM central processors through optional synchronous communications interfaces — IBM's bisynchronous and SNA/SDLC protocols, and Sperry's UNISCOPE protocol.

The Sperry Personal Computer operates at a high speed of 7.16 MHz that benefits computation-intensive business applications such as graphics, spreadsheets and mathematical calculations. A unique switch selection allows the personal computer to operate at a slower speed of 4.77 MHz to emulate IBM if desired.

A monochrome or a choice of two colour display monitors is available. The high resolution colour display provides enhancements over the IBM colour monitor. The medium resolution colour display provides graphics resolution and colour features similar to the IBM PC but at a lower price.

The high resolution colour display of the Sperry Personal Computer doubles that of comparable competitive units while offering a greater array of colours. For example, Sperry achieves a maximum of 256,000 dots of resolution and allows 16 colours compared with IBM's 128,000 dots and two-colour limitation. A maximum of 256 colours may be displayed at one time on the Sperry screen against a 16-colour maximum for IBM.

In the high resolution graphics mode, the Sperry Personal Computer offers the ability to use four different pages, switching screens instantaneously. It is possible to superimpose and change both graphics and text data at the same time.

Sperry's extra-function keyboard has an IBM-lookalike layout for ease of concurrent use, and offers additional features such as clearly defined key captions instead of cryptic arrow; lock key indicators; an extra Enter key placed conveniently next to the numeric pad, and Shift and Return keys in more familar typewriter locations.

The Sperry Personal Computer is available in seven models. Model differences relate to monochrome or colour monitor choice, diskette capacity and fixed disk options. For more information contact Sperry Ltd, Sperry Centre, Stonebridge Park, London NW10 8LS (phone 01-965 0511).

HARD DISCS FROM TANDON

A family of low-cost, high-capacity Winchester-based disc-memory subsystems has been developed by Tandon for "plug and play" memory expansion of the IBM Personal computer by end users, or for integration into new computer systems by OEMs.

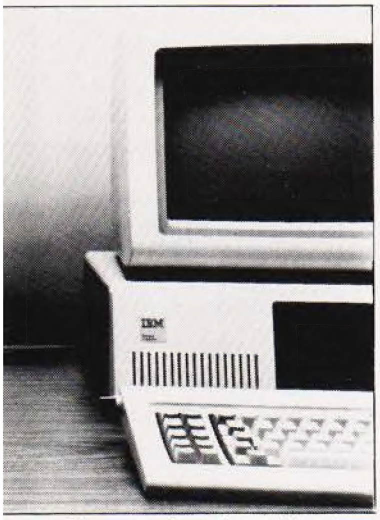
ELF FOR PIXELS

The ELF terminal is a high definition visual display unit with a desk footprint similar in area to an A4 sheet of paper. Its small size is of particular benefit to the executive who requires instant access to data at his desk but who would find the encumberence of the normal sized VDU unacceptable. Its low volume and ease of portability also make the unit ideal for the field engineer.

The ELF has a flat 7" etched screen providing an 80 character line length with flicker-free legibility. The full screen displays up to 24 lines and the 6 x 10 character cell offers a comprehensive range of alphanumerics plus graphics images.

Single action keying doubles text width and height to permit longer distance viewing while further straightforward key strokes command access to the four screen quadrants. In the fully compatible videotex mode, 40 columns with 25 data lines are displayed using the standard 64-character set.

The QWERTY keyboard has supplementary function keys, five LED status indicators and a selectable audible tone generator. The all-up weight of the monitor and the separate keyboard is only 3.6 kg and power consumption is 25 watts from a 240 or 110 V AC supply.

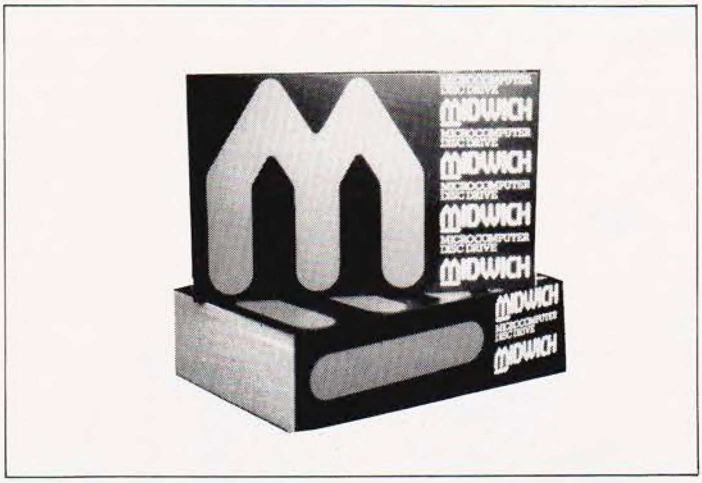


The product line, Tandon Hard Disc Subsystem, is being marketed to both retailers and OEMs through Tandon's industrial distribution network.

Tandon's initial offering is a self-contained subsystem in a compact, desktop configuration tailored for either OEM or end-user applications. This new subsystem is a large-capacity intelligent peripheral providing random-access data storage and retrieval. It is a self-contained package with a controller, a power supply and, for end users, an adapter card for the IBM Personal Computer.

The subsystem for end users, the TM 5110 Series OEM model, comes with an adaptor board to "plug and play" on to an IBM personal computer, and software for IBM DOS 1.1 or 2.0 operating systems. A typical application will be as an enhancement to a single-user IBM/PC with dual 320K flexible disc drives.

Tandon Corporation offers a complete line of full and half-height 5¼-inch floppy and Winchester disc drives, a line of 8-inch half-height floppy disc drives and a new half-inch cartridge tape drive family for Winchester disk drive back-up applications. Further information from Tandon International, 5 Suttons Industrial Park, London Road, Reading, Berks GS61 1AZ (phone 0734 664676).



MORE FROM MIDWICH

In a move to capture a greater section of the retail outlet market, Midwich Computers, East Anglia's leading suppliers of microcomputers, peripherals, components and kits, are launching an own-name range of packaged products which will soon be appearing in shops throughout Britain.

The main packaged item to be sold as a Midwich product will be a non-expandable single 100K disc drive for the BBC Micro. Assembled and packed in a protective polystyrene tray at Midwich's own premises at Rickinghall near Diss, the disc drive is eye-catchingly presented in Midwich's black and green livery. Also available from Midwich very shortly will be general purpose cables and connectors, pre-packed onto card and again printed in Midwich's livery. These will be available to the retailer on wire racks for counter display.

The importance of attractive packaging is high on Midwich's list of priorities as they seek to maintain the standards of service they have already established in the educational and direct mail fields. "As the market matures and end-users purchase from the High Street rather than by mail order, packaging is going to play an increasingly vital role," says David Watson, Midwich's managing director. The Midwich packaging has been carefully designed and produced to a higher than average quality and David Watson is confident that stores will give the products plenty of display space. However, equally aware of retailers' own requirements, he would be willing to arrange for packaging in the stores own-brand packaging if this is preferred.

LOCK UP YOUR BEEB

VEL BEEB-LOCK is a new product designed and manufactured in the UK by Video Electronics Ltd, Wigan Road, Atherton, Manchester M29 0RH. It will synchronise the RGB output from a BBC Micro to a standard interlaced 625-line video signal from a camera, studio, video-disc player or similar equipment.

The BBC Micro can now be used as a low cost alternative to expensive 'professional' television caption or graphics generators. VEL BEEB-LOCK is housed in a small case and is connected to the micro's printed circuit board by a simple wire interface. A unique feature of VEL BEEB-LOCK is the ease with which it can be installed — there are no major track or component modifications required to the BBC Micro — just a few wires to solder onto the board.

The unit even includes a PAL colour encoder and colour genlock system to convert the RGB output from the BBC Micro into a synchronous PAL colour signal which can be combined with the video signal using a VEL Minimixer or any other studio vision mixer. There is also a version available (Model BL.2K) which has its own mix/key board fitted, thus providing a complete system for those users who only require a basic title and caption insertion/fade facility.

Another model, BL.3, is fitted with a MICROPAL board instead of the PAL encoder. This first decodes the reference video signal into RGB components prior to combining them with the RGB output taken directly from the BBC Micro. The final output is a linear RGB signal which can be displayed directly on a suitable linear RGB colour monitor. This approach gives superb quality for direct view applications and will be of particular interest to designers of interactive video-disc systems.

A further enhancement for the BEEB-LOCK encoders will be a 'colour mapping' board which VEL plan to introduce as an update kit in the autumn. This will allow the user to preset up to eight encoded colours selected from several million variations. It will then be possible to have several shades of the same colour on screen simultaneously — very useful for shadow and modelling effects.

UNBEATABLE SOFTWARE PROTECTION

We would like to congratulate Rabbit Software for finally coming up with something the whole industry has been waiting for — copy-proof software. The review copy of their new Commodore 64 arcade game, Troopa Truck which was supplied to CT would defy the most devious of pirates: the cassette contains nothing but leader tape.

Nice to see you've got a sense of humour, Heather . . .

MORE ON MAILING

Canon (UK) Ltd have announced a new facility for their Canon AS100 small business computer which will enable it to communicate with other computers or transmit telex messages via the new One To One service recently announced by Kensington Datacom Ltd. Canon is the first supplier to fully endorse the One to one service on their equipment. The new facility offers users a faster, simpler low-cost method of sending messages via telex, electronic mail or letter.

The computer is connected to a normal telephone line by a compact auto-dial, error-checking modem, and this together with One To One's software program enables messages to be quickly transmitted and received, without the user needing to have a telex line of his own. For small to medium sized companies with a low volume of telex traffic, this can result in an annual saving of anything in excess of £1000 over using a conventional telex machine.

The agreement with One To One, which is estimated to be worth more than £500,000 in retail modem and software sales in the first year, is the result of developments carried out between HRB Ltd — one of Canon's main computer dealers — and One To One. HRB Ltd have been appointed as national distributor for the One To One System on Canon computers.

The AS100 16-bit small business computer is available in monochrome or colour versions with high resolution graphics capabilities. Memory sizes of 128K to 512K and a choice of floppy disc or Winchester disc options from 1.2 Mb to 8.6 Mb make it a versatile business system. There is a wide range of applications from some of the leading software houses for word-processing, spreadsheet, financial modelling and database, in additon to financial systems and vertical maket add-ons, running under CP/M86, MS-DOS or B.O.S. operating systems.

A networking system supporting up to 64 workstations is available, supported by 20 Mb, 40 Mb or 80 Mb disc capacities, and the range of printers includes daisy wheel and colour graphics ink jet models.

Canon are of Waddon House, Stafford Road, Croydon CR9 4DD (telephone 01-680 7700).

BELIEVE IT OR NOT . . .

. . . but Uncle, Sir Clive Sinclair has just been voted by company directors in the British electronics industry as having the UK electronics company which, "through its achievements in the past year, has most enhanced the image of the industry as a whole". Apparently none of the people entitled to vote were on the interminable, thousands-strong list of customers waiting for a QL: a machine which was 'launched', may we remind you, on January 12th, and which still doesn't have its ROM finalised, we understand.

The real joke is that one of the judging critieria was customer services. Hanging on to customers' money while still developing your product does not strike us as being much of a service.

HUSKY BY NAME . . . AND NATURE

The Husky Hunter is a professional handheld microcomputer featuring exceptionally large memory capacity — up to 208K byte of CMOS random-access memory together with a 48K byte ROM-based operating system — and rugged packaging for use in the most stringent environments. Together with CP/M emulation, Husky Hunter also contains RAM-disc facilities which enable conventional CP/M disc files to be stored and used in exactly the same way as in a conventional desktop computer.

These features mean that the most successful microcomputer software packages can be used in the handheld environment for the first time ever. Husky Computers believe that this critical technical innovation will allow handheld computers to be used in widespread applications and will effectively remove restrictions from the use of handhelds for practical commercial and professional tasks.

As well as its unique CP/M and disc emulation features, Husky Hunter is also extremely robust and has a large LCD display screen that can be used for alphanumeric or graphic displays. The system also contains full graphic support through a built-in BASIC interpreter.

The ruggedness and portability of Husky Hunter mean that it is ideally suited to data-capture and computing applications in all-weather outdoor environments, where it can replace traditional pencil-and-paper methods in areas such as meter reading, sales order entry, surveying, and remote data gathering.

In the general business environment, Husky Hunter's ability to run large, complex programs such as Supercalc means that it can be used by financial managers to prepare and update budgets even when they are away from their desks. In addition, a wide range of applications are opened up

in areas such as work study, maintenance scheduling, or licensed-trade stock control.

Husky Hunter measures only 216 x 156 x 32mm — no larger than an A5 paperback book — and weighs just over a kilogram. The diecast aluminium alloy case makes it rugged enough for the most heavy-duty applications — even against accidental immersion in water.

The ruggedness of Husky Hunter extends to its data-storage capabilities. Memory contents are protected by a two-tier fail-safe battery system, and all the keys on the 57-key QWERTY-style keyboard can be reallocated by software to prevent data being accidentally erased by an incorrect key operation.

Husky Hunter is easy to program using a built-in BASIC interpreter, and its CP/M compatibility means that it can be used with thousands of commercially available software programs and other languages. In addition, special data-collection packages have been developed for use in specific application areas.

Advanced features of Husky Hunter BASIC include extensive graphics commands such as five character sets, circle generation, box drawing and infills, plus multi-octave sound generation. Additional facilities include random and sequential file handling, a real-time clock/calendar, and a word-processing-style screen editor that covers the full 96-character virtual screen width with a built-in scroll facility.

A built-in RS232 interface and a choice of communications protocols, including IBM 2780, mean that Husky Hunter is compatible with virtually all mainframe computers and with many other micros. In remote data-capture applications, direct software-controlled communication can be carried out via normal telephone lines to download stored data for subsequent processing.

A range of peripherals, accessories and options is available which allows Husky Hunter to be used as part of an integrated data-capture and computing system. These include a disc drive unit, a bar-code reader and printer, a dot-matrix printer, an acoustic coupler and a range of modems for synchronous and asynchronous communications.

Husky Computers Ltd are at PO Box 135, 345 Foleshill Road, Coventry, CV6 5RW (phone 0203 668181).

THE FUTURE IS HERE

The latest business micro from Future Computers is unique for its price in having a built-in cartridge tape. Note even the computer's main rivals, the IBM PC/XT and Sirius, can match this feature of the all-British FX30T.

Two versions of the FX30T are available, offering a choice of hard discs with 20 or 40MB capacity. These hard discs can store huge amounts of data and retrieve it at high speed, making them much better for serious business use than the floppy discs used by many business computers.

With so much valuable data stored on a hard disc it is essential to keep back-up copies of the data in case of accident. Usually this is done by copying the data on to floppy discs, but this can mean using 25 or more floppy discs to copy a 20 MB hard disc. This is time-consuming, error-prone and costly.

The FX30T has a built-in cartridge tape unit. This uses tapes which each hold 20MB of data, making it easy to produce back-up copies of the hard disc. The system allows the whole disc to be copied, or just selected files, such as those updated since the last back-up copy was made.

Despite being at the forefront of technology the FX30T is modestly priced. The version with a 20 MB disc, the FX30/20T, costs £4150 (plus VAT) and the 40 MB version, the FX30/40T, costs £4750 (plus VAT).

As well as the hard disc and tape unit the FX30T comes with 128K of memory (expandable to 1MB), keyboard, and 12" screen all housed in a stylish case. It uses a powerful 16 bit microprocessor (an 8088 running at 8 MHz), making it faster than most of its rivals. A wide range of business software is available for the FX30T's CP/M-86 and Concurrent CP/M operating systems and its optional MS-DOS operating system.

Like all the micros in the Future Computer range, the FX30T has a networking interface built-in as standard. This means it is just a matter of plugging the computers together to get them to talk to each other. This is particularly useful with the FX30T because it allows cheaper computers in the FX range to share its hard disc and cartridge tape unit.

The FX30T is available now from Encotel Systems, the distributor for Future Computers, on (01) 680 9640.

PLAYBACK FROM PANASONIC

The RQ-8100 program recorder from Panasonic accumulates data quickly at a storage rate of 1,200 bits/sec, and provides nearly 500 kilobytes of information on a C-60 cassette — more than most floppy discs. A remote jack means that the RQ-8100 can be connected to many brands of computers, therefore permitting automatic start/stop commands, sent from the keyboard.

Locating specific programs and data poses no problem — the fast forward and rewind functions are not affected by remote control, so the FF/CUE and REW/REV buttons can be used to quickly locate specific programs and data; a phase selector corrects errors while loading commercial software and the monitor capability provides confirmation of signal transmission during loading and saving.

Designed with emphasis on ease and simplicity, the recorder weighs 600 grams without batteries, and measures 119 mm by 40 mm by 202 mm. Other features include a carry handle; one touch recording; follow-up recording; auto-stop; lockable pause control; tape counter; tone control and built-in condenser microphone.

Economically powered, the unit operates on both mains (with optional AC adaptor) or battery (four 'R6' size batteries). Selling at a recommended retail price of £55.50, the RQ-8100 is now available through Panasonic's authorised dealer network.

CLEAN UP YOUR AC(T)

A new, truly portable Mains Adaptor Filter, the DS23384, designed for easy use and flexibility in the modern office and laboratory, is now available from MPE-Dubilier, leaders in the field of FRI/EMC suppression filtering. Plugged into any 13 amp mains socket, the equipment or system to be protected can then be used from the adaptor, with no wiring necessary.

With integral overload cut-out, 'mains-on' indicator and full transient protection, the range also includes two and four way versions. For further details contact MPE-Dubilier Ltd, PO Box 11, Hammond Road, Kirkby Industrial Estate Liverpool L33 7UL (phone 051-547 2600).

HERE COME KAYPRO

A new range of portable personal computers is being introduced in the UK, having taken the American market by storm last year. Kaypro computers are claimed to offer very high levels of performance at a price considerably lower than similar machines. They are being offered not only with all the necessary hardware and software included, but also special training programs are being loaned free of charge from the dealers to customers using a computer for the first time.

The Kaypro II is a compact, portable microcomputer which is designed for home and small business use. It costs £1,095 plus VAT, which includes the processor, twin disc drive, 9" monitor and keyboard, plus software to carry out word processing, correct spelling, financial spreadsheet, database management and programming functions. It has recently been voted portable computer of the year by leading US and European journalists in *Chip* magazine.

The Kaypro 4 augments the best features of the II but with double-sided disc drives allowing storage ot twice the amount of data for larger businesses and professional applications. It costs £1,595 plus VAT, again including a complete package of hardware and software.

The Kaypro 10 combines the speed, storage and power of a 10 megabyte hard disc drive with high-resolution graphics capability. It costs £2,295 plus VAT and is intended for business and professional users requiring an extensive database, such as insurance brokers, accountants, estate agents, travel agents, and stock brokers.

The core of every Kaypro computer is the popular Z80 microprocesor with a 64K RAM memory. Any of the thousands of programs available on CP/M software can be used utilising the integral floppy disc drives. The Kaypro II provides up to 400,000 bytes of storage, the Kaypro 4 up to 800,000 bytes, while the Kaypro 10, which is equipped with both floppy disc and Winchester hard disc drives, stores nearly 10.5 million bytes of information.

Each model incorporates a large 9" monitor, which displays up to 24 lines, 80 characters wide in sharp green phosphor letters, adjustable for brightness. Also included in the price is a 72-key professional-quality typewriter-style keyboard with 20 programmable keys plus a 14-key calculator-style keypad for easy entry of statistical data. Input and output connections for communications with other computers and any printer are also provided.

All Kaypro computers come in robust, steel cases, so that they can be folded up and carried home for personal use, placed in the car boot or under an aircraft seat on business trips.

Kaypros come complete with a package of 8-10 ready-to-run programs for the office. These include word processing, spelling checkers, financial spreadsheet, database management, and programming languages.

Kaypro are at Elm House, Elmshott Lane, Cippenham, Slough, Berkshire.

WINNING WAYS

The best word processor — according to a survey among users of 39 different models — is the RDS 200, marketed in Britain by Data Logic Limited. This was the finding of a survey carried out in the United States by Datapro Research Corporation. Datapro is an independent market research organisation specialising in the area of office automation.

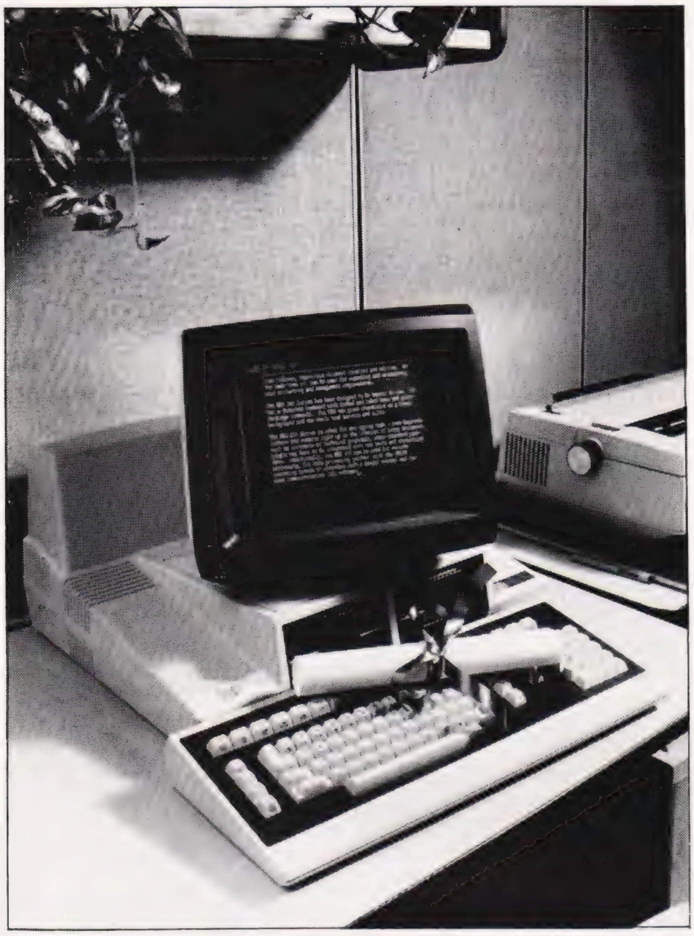
The RDS 200 came top among users for overall satisfaction. There was not a single user, according to the survey, who would not recommend it to someone with similar word processing needs. RDS 200 users also rated it the best machine in two other categories: ease of operation and reliability of performance. The RDS 200 is designed for ease of use: it uses plain language commands, a detached keyboard and a VDU head that swivels and tilts. It can also be used for records and data processing and data communications. The basic RDS system costs from £5,445.

For more details, contact Data Logic Limited at 29 Marylebone Road, London NW1 (phone 01-486 7288).

GETTING DUPED

Leading audio equipment manufacturer Soundcraft has launched its new high speed cassette duplicator. Aimed at the professional studio, computer and educational markets, its extremely fast duplicating speed of 31 inches per second (which is 16½ times faster than real-time copying) enables any 60-minute cassette to be duplicated in under two minutes. A dynamically-controlled manual or automatic rewind facility takes a C60 back to the start in less than 35 seconds.

The CD 201's modular design allows a limitless number of

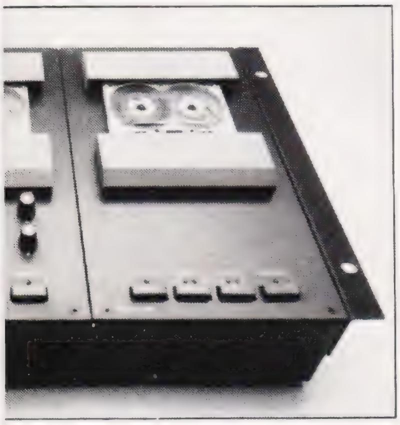


tapes to be copied by daisy-chaining individual slave units to the master unit. Only additional power supply units are required for each group of four slaves added. Its switchable chrome/ferric tape type facility means that a mixture of chrome and ferric cassettes can be copied simultaneously because the correct bias parameters are pre-set and switched electronically for each tape type.

Maximum reliability has been attained by keeping moving parts to a minimum. The only moving mechanical component is the single pivot floating head mechanism — so mechanical wear is negligible. Both the reel motors and capstan motor are direct drive and contained within a single cast aluminium transport block. Crystal-referenced, phase lock loop capstan motor control and dynamically controlled spool motor tensions make wow and flutter insignificant. Soundcraft claims that with a deviance rate of only 0.1%, the CDE 201 has the lowest wow and flutter specification in the market.

Features include a high density, glass bonded ferrite head with a stereo four-track configuration that copies both sides of the cassette at one pass. The head mechanism is logic-controlled, locked into position when engaged and with dynamic back tension ensures excellent tape to head contact.

For anyone thinking of setting up business in the expanding tape duplication industry, the CD 201 costs £1290 (excluding VAT). Further details may be obtained from Soundcraft Electronics Ltd, 5-8 Great Sutton Street, London EC1V 0BX (phone 01-253 6988 or 01-251 3631).



TANDATA FOR IBM

Tandata Marketing is now able to provide IBM PC software for use with its smart modems (Tm100, Tm100x or Tm200), or its Td1404 and Td1616 controllers when used as modems, to provide Prestel terminal emulation within the capabilities of the IBM PC or any PC from 64K single disc monochrome upwards. 128K is required to take full advantage of autodial and some other features.

The software — including a graphics chip for full Prestel graphics — costs £150, and a modem-to-micro cable terminating in a 25-way D plug (female) costs £15 (both prices exclusive of VAT). Full information and the software itself is available from Tandata Marketing Limited, Albert Road North, Malvern, Worcestershire WR14 2TL (phone 06845 68421).

The IBM PC software is the latest in the fast expanding range of Micropacks now available from Tandata — software is already available for BBC Model B and Apple Europlus/IIe, and will shortly be available for the Commodore 64, Commodore PET range, VIC 20 and CP/M micros.

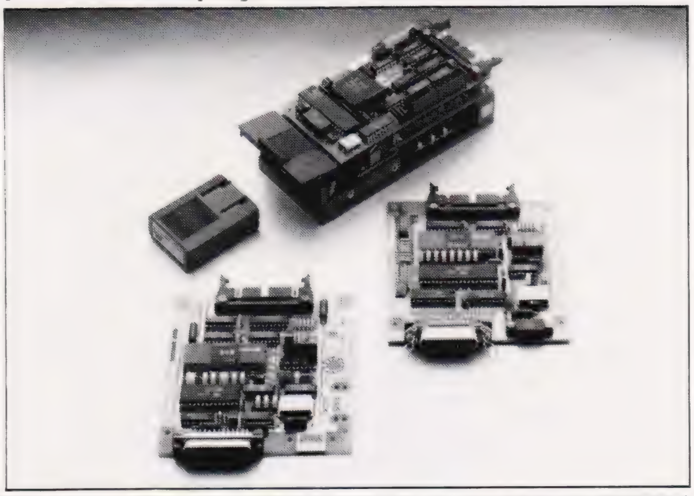
BUBBLING UNDER

Now available with an RS-232 serial interface card or an IEEE 488 parallel interface card, the IBS 1 Megabit bubble cassette system is a new kind of exchangeable memory that provides reliable operation in the most demanding environments. A small, lightweight, maintenance free unit unaffected by dust, dirt and vibration, it has obvious advantages over disc, magnetic or paper tapes.

With error rates of 10^{-13}, bus parity, error detection and correction functions, the system provides extremely high data integrity. The capacity of the system can be easily expanded to 512 Kbytes, and with achievable high speed access times of 2.6mS, this is the fastest, most reliable exchangeable non-volatile memory system available. The unit costs are as follows:

IBS C128 Bubble
Cassette £278
IBS H100 Master Unit . . . £245
IBS HX1 Slave Unit £117
IBS H300 Master Unit . . . £197
RS232 Interface for
above units . . . £169

Further information is available from IBS (Milton Keynes) on 0908 568192.



MEMOTECH BENCHMARKS

We were pleased to receive a letter from Memotech congratulating us on our review of the MTX500 (April '84), which modesty forbids us from reprinting. However, there seems to be some confusion over the value of Benchmark 8, which Memotech think should be 4.3 seconds rather than the 42.7 seconds we published. The reason for the discrepancy, of course, is that we took our timings over the same number of iterations as all the other benchmarks, while some other magazines take benchmark 8 over 100 iterations to make it less tedious to run. Hopefully this clears up any doubts among readers that the MTX was doing trig calculations on an abacus. (In any case, as we pointed out in the article, benchmarks are a silly way to compare computers).

NEXT MONTH

Computing today

JULY ISSUE ON SALE JUNE 8th

CONJURING A GENIE

Audiogenic are best known for their range of arcade games for home computers such as the Commodore 64, VIC 20 and Spectrum. Now they've diversified a little and have moved into more serious applications. Next month we'll be looking at their database system for the 48K Spectrum, called Data Genie, which brings the concept of pop-up menus, or windows, from the heady heights of the Lisa and Macintosh into the realm of Sinclair users. Go on record with the July *Computing Today.*

COMMODORE 64 BASIC EXTENSIONS

With practically every useful hardware function on the Commodore 64 being controlled by a series of PEEKs and POKEs, it's no wonder that BASIC extensions are springing up everywhere to provide new keywords. Why, we're doing a series on this ourselves. If you don't want to tailor your own BASIC and would rather buy an off-the-shelf cartridge, though, then the next issue of *Computing Today* is essential reading. We'll be comparing a number of BASIC bolt-ons and seing how good they are.

INTERFACING TO THE ORIC

The other day we had one of those Angry of Mayfair telephone calls. "I've seen a lot of rubbish published in computer magazines about interfacing hardware to the Oric, and I want to tell people how it should be done". When the speaker is one of the men behind the Byte Drive disc system for the Oric, you sit up and take notice. And you get him to write an article, and it appears next month.

A MORE COLOURFUL DRAGON

People are constantly coming up with unusual facts about the Dragon's graphics modes, and in the July issue we'll have yet another. Want a few more colours in Mode 3? We'll show you how to mix up a whole new palette.

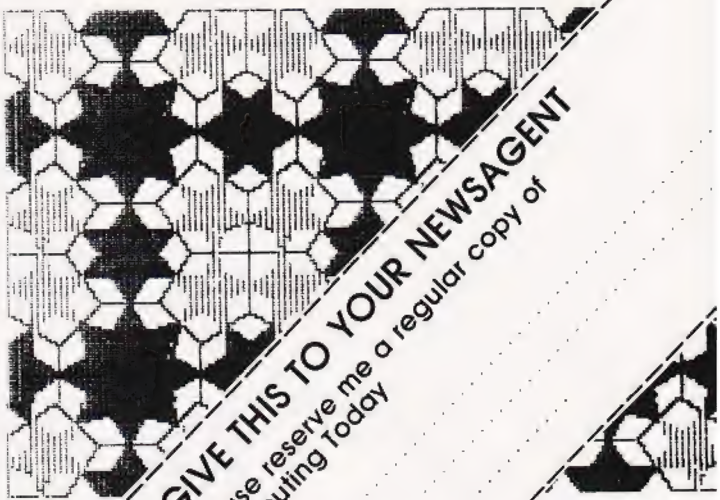
Articles described here are in an advanced state of preparation but circumstances may dictate changes to the final contents.

TANDY MODEL 4P

Some months ago *Computing Today* ran a feature on three new computers from Tandy. One of them, the Model 4, was a combined Model III and Model 4 machine, running both new Model 4 software and converting to a Model III for total compatibility. Now, amazingly, this two-in-one machine has been transformed into a portable version, the Model 4P. Our business computer expert will be putting it through its paces next month.

ART AND GEOMETRY

The Byzantine Empire, among other things, produced some outstanding examples of how geometrical concepts can be applied to produce artistic decorative patterns in architecture. Designs produced on a grid of regular polygons obviously lend themselves to the techniques of computer graphics, and next month's *Computing Today* shows how some interesting effects may be obtained on the BBC Model B and Apple microcomputers. Dabble in the art of the East with the July issue.



GIVE THIS TO YOUR NEWSAGENT

Please reserve me a regular copy of Computing Today

NAME ....

ADDRESS ....

POSTCODE ....

ALPHATRONIC PC

Simon Dismore

Triumph-Adler, the German office automation manufacturers who created the successful range of Bitsy word processors, are about to launch their new eight-bit Alphatronic PC on the UK market. Computing Today examines the prospects for a new machine in an already overcrowded market.

First impressions of the PC are that it is an attractively styled machine with a distinctively European (rather than American or Japanese) feel. Underneath the covers, it looks much like any other single-board computer, with economically-designed circuits, power supply and interfaces. The review machine was presumably a pre-production model, as evidenced by the hand-soldered capacitors on some of the RAM chips, which are certainly Not A Good Thing (as users of other manufacturers' products have discovered to their dismay).

The machine is manufactured in Japan. Triumph-Adler claim that this gives users the best of both worlds: "Teutonic attention to design combined with Japanese quality control". The PC's specification certainly matches that of a number of Japanese machines: Z80A processor with 64K RAM (of which 4K is used for memory-mapped video), Microsoft BASIC in ROM and an Intel 8251 chip to take care of communications.

In all, a workhorse machine with a fairly conventional personality compared with home-grown products like the Beeb. Anyone with experience of one of the smaller Japanese business machines should find that the PC presents few surprises. At around £347 for the entry-level cassette version, it represents reasonable value for the small businessman, and can be expanded to a twin-drive CP/M system.

Who would buy it? Triumph-Adler call it a home/career/business machine, but we felt that its philosophy was far more aimed at business than home.

HARDWARE FACTS

The keyboard has a business-like feel and should stand up to heavy pounding, and the calculator keypad and function keys (six, which can be shifted to produce 12 combinations) will be welcomed by business users. There was no utility to configure the function keys from CP/M, but one should (hopefully) be available when the machine is launched.

The bad news is that several keys are in the "wrong" place: the TAB key is on the right-hand side, underneath the Return key; the CONTROL key sits where the TAB should be, and so on. It is hard to see why this has been done, as the PC literature shows a keyboard with the correct layout. Worst of all, there is no Backspace key, which is essential for most applications. Triumph-Adler advised us to "use cursor left and delete" instead, but it seemed more sensible to reconfigure one of the function keys as a backspace (KEY 6,CHR$(8)).

We also found that the PC got very confused about which shifts were in operation: press the Lock key a few times and the Graph LED lights up! Any users who find this happening to them should ask for a replacement machine immediately — this is not the sort of fault which goes away with time.

There was no £ symbol on the keyboard or screen of the review machine. Most businesses find this little character rather useful, and Triumph-Adler now tell me they will be providing this on new machines coming into the UK.

There are outputs for RGB and (monochrome) composite video monitors, and Triumph-Adler supply a useful UHF modulator with the system which converts the RGB signal for use on a standard TV receiver. We found that the signal strength from the modulator (which takes its power from the RGB signals) was quite weak, so it was difficult to get good colours on a TV. The character shapes are very good, and if reverse video (black characters on white) is used it is possible to get a reasonable 80 column display on a TV — not good enough for intensive work, but probably sufficient for occasional word processing, and certainly better than the BBC Micro in 80-column mode.

Unfortunately, whenever a deletion is made on the screen there is a strong flicker (presumably while the video memory is being updated) which is at its worst when using reverse video. It was not clear whether this was a defect of the machine, or just the ROM BASIC. In any case, any business user would soon find an RGB monitor essential for 80-column work, which makes reverse video unnecessary.

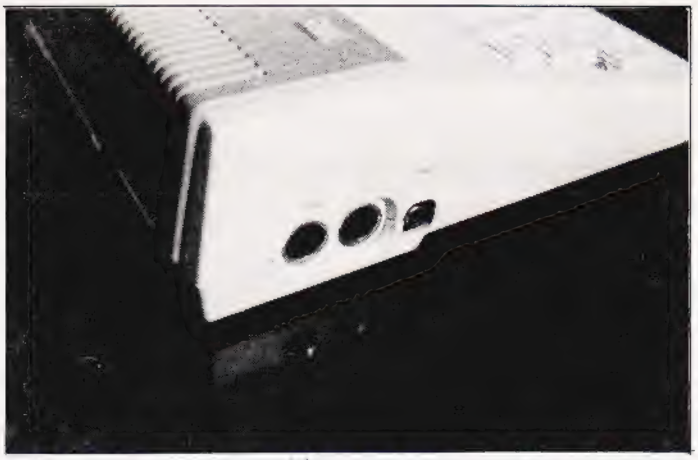
Hobbyists would not be impressed by the limited graphics capabilities of the PC. Eight foreground/background colours are available, but the 160 by 72 pixel resolution (80 by 72 in 40-column mode) would be inadequate for all but the most simple games and there is no facility for user-defined graphics. Triumph-Adler say that a high resolution graphics board will be available for the home market, but would not give any indication of price or delivery. The combined price of PC with graphics board might put the system out of reach for most home users.

LANGUAGES

The machine has the standard Microsoft BASIC in ROM, and disc drives come with Microsoft Disc BASIC ("DOS"). In each version programs can be edited using the cursor keys in full screen mode, and (apart from the irritating flicker on deletions) the CRT display is quite adequate for program development. Like many Japanese machines, the function key labels are displayed at the foot of the screen for easy reference.

Software can be loaded from cassette (Kansas City I/O interface through a DIN plug), PROM cartridges or disc. The 16K PROMs fit into a concealed socket on the top of the machine. Triumph-Adler will be launching their first cartridge (a custom Typing Tutor) in the near future.

Discs connect to the system through the I/O bus, which is brought out to a rugged connector on the back of the PC. Triumph-Adler market 5¼" drives for the PC, which offer 320K storage (unformatted) at £330 for the first drive (including controller) and £270 for the second. Most 5¼" drives are slow, and these were no exception. However, a single drive configuration for about £677 (plus £29.50 for CP/M) never-



theless represents reasonable value for money.

IN USE

The Alphatronic really cannot be described as a home system in its current configuration with low resolution graphics. Triumph-Adler say that they treat this market very seriously, and are putting a lot of effort into software development particularly in the interactive education field. We did not have the opportunity to test any of this new software, described by T-A as "Education with a big E", but the games that we examined were truly atrocious: years out of date and full of misspellings. Perhaps Triumph-Adler will improve these before they launch the system, as they made a number of references to other manufacturers who they said were "claiming an awful lot, and delivering rather less".

The release documentation was still being prepared, but we were able to see a well-produced dealer pack listing many items of software which will be available for the PC, including **Crystal** (a cassette-based business system), **Alias** accounting software, the **Milestone** project managment system and a large number of spreadsheets, including **Masterplanner, Supercalc 2** and Micropro's **CalcStar.**

FACTSHEET	Triumph-Adler Alphatronic PC
CPU	Z80A
Clock speed	4 MHz
ROM	32K (24K BASIC, 4K IPL)
RAM	64K
Language	Microsoft BASIC
Keyboard	QWERTY standard plus numeric keypad and programmable function keys (total of 85 keys)
Display	80 or 40 column text with user-defined scrolling window of one to 24 lines. 160 by 72 or 80 by 72 pixel graphics. Eight colours. RGB, Composite and UHF video (with adaptor supplied).
Cassette	Kansas City I/O
I/O	RS-232C interface Centronics port Bus I/O for floppy discs or expansion
Weight	3.5 kg
Options	Disc drives (one or two 320K 5¼" units) Microsoft disc BASIC CP/M 2.2 File server *(under development)*; 10M Winchester disc (8.2M available to users), 20M tape streamer, SDLC network using 75 ohm coaxial cable with data transfer at 78K/second
Prices	Alphatronic PC £347 plus VAT Discs: First unit £330 plus VAT Second unit £270 plus VAT File server £5735 plus VAT (NYA)
Manufacturer	Triumph-Adler (UK) Ltd, 27 Goswell Road, London E1M 7AJ (phone 01-250 1717)

As a business system, the PC is interesting more for its price than for any special features. There will not be a Prestel or Teletex interface, but Triumph-Adler have prototyped an interface to the **Micromite** file server. This allows up to 254 micros to share information on a Winchester disc unit, and would certainly be useful for the larger organisation where a number of users need access to the same information.

A typical network configuration with a 10M hard disc and a 20M tape streamer (for making back-ups) would cost around £5735, plus the (unspecified) cost of upgrading each PC on the network. At the moment, each PC must have at least one disc drive in order to work on the network, but Triumph-Adler hope to have a disc-less version "in due course". If the network interface is inexpensive this disc-less version would make the file server more attractive than fitting single drives to 20 machines (or twin drives to 10 machines). Triumph-Adler were, quite rightly, making no promises about what are still only prototypes — but it will be interesting to see what they come up with.

THE FUTURE

The Alphatronic PC faces stiff competition when it is launched. The Japanese will be creating a lot of pressure with their eight-bit MSX standard machines, which will offer the graphics that home users want, and there are already some very competitive small business configurations appearing. For example, the new Sanyo MBC-550 (a 16-bit machine with 128K RAM, MS-DOS and one 160K drive) is retailing in London for £669 (all prices in this article exclude VAT). And, of course, there is the Sinclair QL . . .

Is it worth buying an eight-bit system when most businesses are rushing to convert to 16-bit? It depends what you want to do, and how much you want to pay. The price of CP/M-80 software is falling, and will probably fall even further once MSX is launched in the UK. Dealers, manufacturers and many users now have considerable experience in programming and supporting Z80-based applications. By contrast, 8088/8086 experience is rarer, and software is more expensive. Remember that, to the user, there is very little difference between running, say, WordStar on a 16-bit or eight-bit system. Big spreadsheets and large databases may run faster — but the first-time purchaser is unlikely to be working with vast amounts of data in any case.

Documentation and support will be critically important. CP/M is not an easy operating system to learn, and at these prices dealers will not have much of a margin for helping novice users. Some manufacturers already include an independently published book on CP/M with their systems, and Triumph-Adler might do well to increase the price of CP/M to include some training documentation, or at least ensure that their dealers hold stocks of an introductory text.

The PC will be sold through existing Alphatronic Micro dealers, who will presumably provide the highest level of support. Triumph-Adler hope to distribute the product through some of the new microcomputer chains and the retail multiples, but did not wish to mention any names at this stage. Look for dealers who can give either a discount on software (in which case you must learn it yourself) or can show a good track record on support, preferably with other Alphatronic products.

CONCLUSIONS

On balance, we vote a cautious "yes" to the Alphatronic PC for the few small business applications that don't require a backspace key. If Triumph-Adler decide to fit one on future machines, and can eliminate the irritating CRT flicker under BASIC, we would be more optimistic. A low-cost (or free) accounting package would also be attractive, as the strange keyboard layout makes serious word processing unlikely. Paradoxically, the danger of buying this machine is its low price, which is the main selling feature of the system. It is terribly vulnerable to competition from low-cost Japanese business systems and expanded versions of home machines. If you don't need a machine right now, wait to see what the next six months brings.

PHOTOCOPIES

A Computing today **Reader Service**

To take advantage of this simple service, just fill in the required information and send it (or a photocopy) to:

COMPUTING TODAY Photocopies,
No. 1, Golden Square,
London W1R 3AB.

together with your money and we'll do the rest!

Lost and can't be replaced? Well, if you've lost one of the early issues that cannot be replaced from our stocks of backnumbers, all is not quite lost.

If you know the article name and the issue it appeared in, we can supply you with a photocopy for the miserly sum of £1.50 including postage and packing. If you're stuck for the month, try checking in our annual index which is included in the December issue each year.

CT JUNE '84 CT PHOTOCOPIES

NAME
ADDRESS

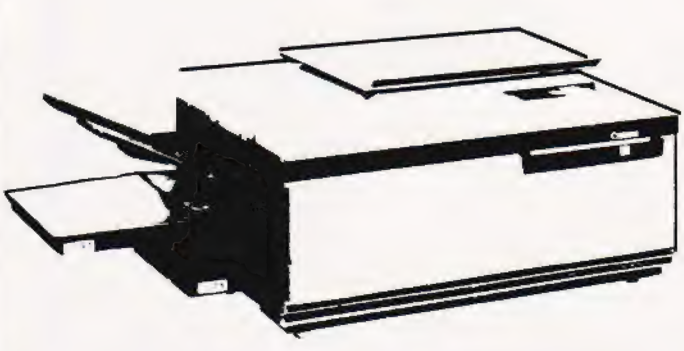
POSTCODE

Please send me Photocopies of the following items

ISSUE	ARTICLE	PAGES

At £1.50 each, I enclose £ ....

Cheques and Postal Orders should be made payable to ASP Ltd



SEEING THROUGH

THE BEEB

Jamie Clary

If the work of art on the cover impressed you, you can do the same thing yourself for a relatively modest outlay. Get hold of a MicroSight system and become a REM-brandt.

Anyone watching the Winter Olympics this year could be forgiven for missing the new range of special-effects which were being used to display snow-bound competitors in a way that the organisers would never have thought possible. Some would have noticed, however, that the operator of the latest special effects module had probably bitten off more than he could chew, as at one point in the proceedings the ever-present stopwatch display suddenly appeared at several different places about the screen — simultaneously!

Although the system being used by the Beeb was an example of some of the very latest in image processing technology, several relatively low-cost systems have been devised which, as peripheral elements to a host microcomputer, are said to offer the user a chance to at least *experiment* with the relatively new art/science of image processing. Such a system will allow the user to seize a single video frame via a camera, transfer it into the screen memory of the host micro, and manipulate the image using standard memory-manipulation techniques. One such system, MicroSight, from Hertfordshire-based company Digithurst, is a video-camera plus interface plus software combination which, for a very modest outlay, offers a rudimentary vision system for use with a variety of micros in home, laborarory, and industrial environments.

Our system was supplied complete for use with the BBC Micro, although putting Microsight into operation and achieving acceptable results required some determination and not a little imagination, as I discovered during the review. Before discussing this in any detail, however, let us first consider the complete package as it appeared one morning at our offices . . .

THE SYSTEM

Our system consisted of a fairly inexpensive-looking, monochrome video camera, complete with a small but adequate tripod, the Microeye interface which provides the link between camera and micro, and two cables — a 20-way ribbon cable terminated with standard insulation-displacement connectors to couple the Microeye to the Beeb's user-port, and a metre of co-ax cable terminated with good-quality

BNC connectors to link the camera to the interface. The standard MicroSight system is, of course, supplied complete with a selection of disc-based programs, and this allows the system to be up and running almost immediately. However, I decided to investigate each element in a relatively methodical and systematic fashion. To this end, I chose to ensure that the camera was in working order.

THE CAMERA

An ITC Ikegami video camera acts as the front-end to the system, and although its appearance prompted adjectives such as 'functional' and 'inoffensive' to spring to mind, it is capable of producing a very accurate image with its output fed directly to a video monitor. Its aperture can be adjusted from f1.4 to f16 in eight detented steps, and it is sensitive enough to produce good images even under lighting conditions normally considered poor ie a 14' by 20' by 12' room illuminated by a single, shaded, 100W light bulb. The lens supplied with the ITC focuses down to a range of one half-metre, which was a little unsatisfactory as many of the more useful applications for the MicroSight system (pattern recognition and so on) would generally require close-up or macro facilities.

Our camera had a small motor slung to its underside and this, in conjunction with a thin, cardboard disc containing red, green and blue filters, enables the production of 'colour' images. This feature I shall expand upon later, but believe me, it's pretty clever if one remembers that the system uses only a monochrome camera!

THE MICROEYE INTERFACE

The video camera and the micro are effectively separated by the Microeye interface, which to the outside world appears as a metal enclosure measuring 25 cm by 18 cm by 8 cm. Mounted to the front panel is an illuminated on/off switch, with four I/O connectors situated at the rear of the enclosure. A BNC socket provides an entrance for the raw video signal, while a 15-way D-type connector joins the interface to the micro via the 20-way ribbon cable. A nine-way D-type connector is used to deliver control information to the motor slung

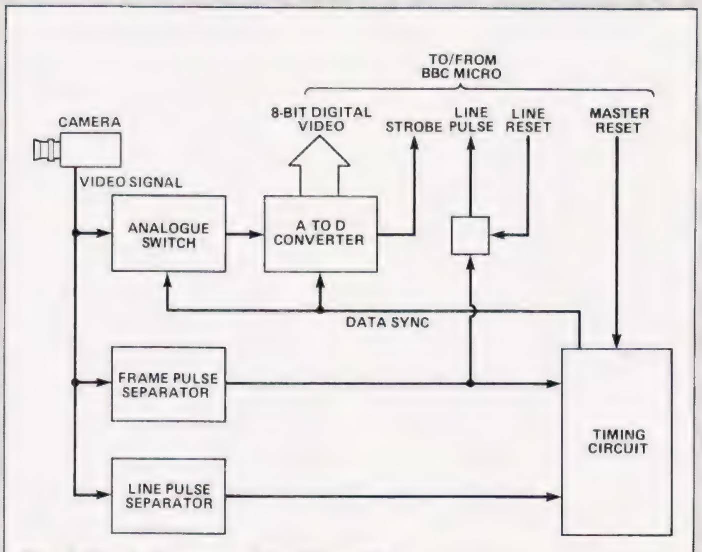


Fig. 1 Block diagram of the MicroSight.

beneath the camera, and finally a three-pin socket forms the 240 V supply entry point.

The technique used to convert the video information into parallel digital form is outlined in the function diagram shown (see Fig. 1). The diagram, taken from the MicroSight manual, illustrates how the video signal is separated into two basic components. Timing information — the line and frame separation signals — are processed and used to control the flow of a second component (beam intensity, effectively) via an analogue switch. A knowledge of these processes is not necessary to use the system, but it was quite reassuring to learn that such details were available if required.

SOFTWARE

Before describing the software in detail, we should briefly consider how this system, and others of its genre, represent images. Monochrome television pictures are constructed from a large number of very small dots. By varying the intensity of each dot it is possible to represent monochromatic shades, hence a definite picture consisting of many shades can be produced.

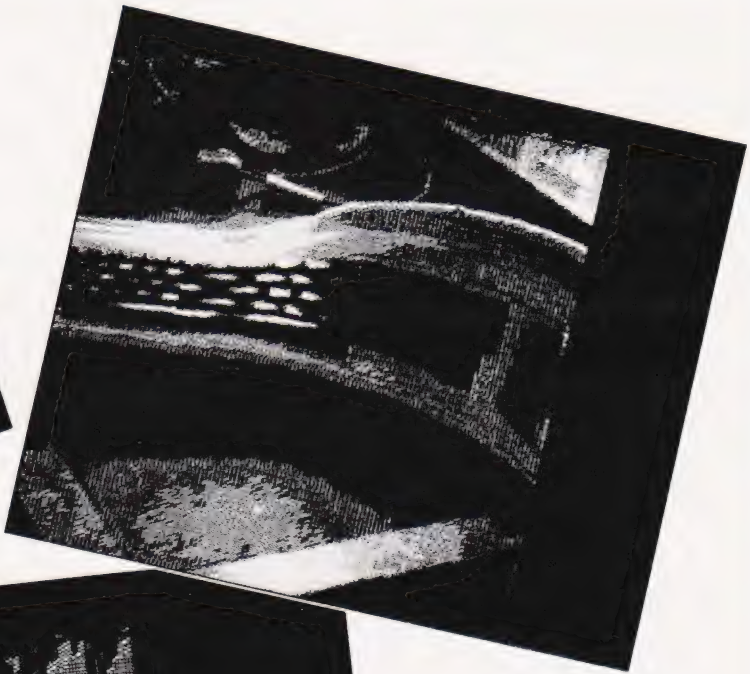
In a similar way, it is possible to represent images upon a memory-mapped screen. Although the ability to vary the individual pixel brightness levels is confined to a relatively small number of micros (it is often more useful to use such attribute space for different colours), it is the case that different colours displayed upon a monochrome screen appear as varying intensitires of grey — the so-called grey-scales. This technique is used with the MicroSight system. The BBC Micro can display eight colours in Mode 2 (flashing 'colours' ignored!) and is therefore capable of displaying eight levels of grey — simple! The drawback is that one has to display such images on a monochrome screen, otherwise the resulting picture, although interesting, appears as a jumble of different colours!

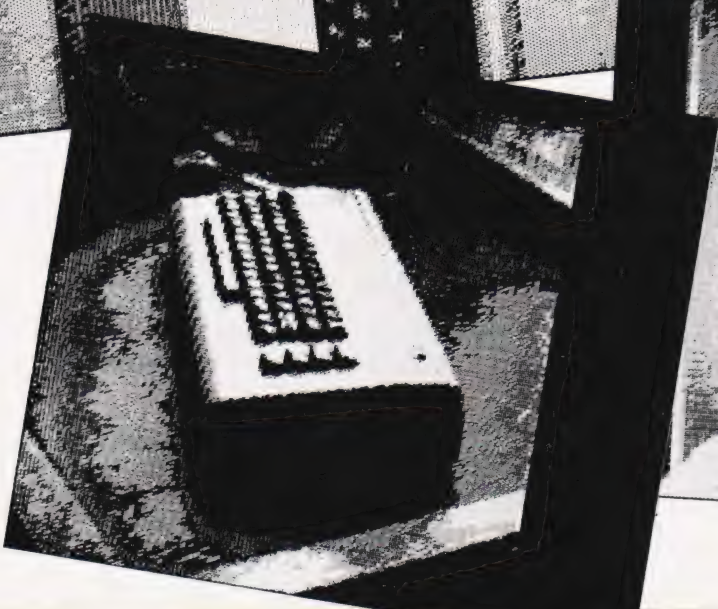
Although the software supplied with a peripheral of this complexity should ideally be capable of exercising every facility that the device can offer, there is an almost infinite number of tricks that one could play with an image once it has been digitised. For this reason, it seems that Digithurst have deliberately supplied a minimum of bundled software — enough to allow images to be captured and subsequently stored on disc, but little which will exploit the system's real potential. This is best emphasised by the fact that the manual for the basic MicroSight package contains not only a full source listing of every routine supplied, but complete, detailed commentaries enabling — and effectively encouraging — users to fiddle with the code in order to obtain a particular facility.

DISC-CONTENTS

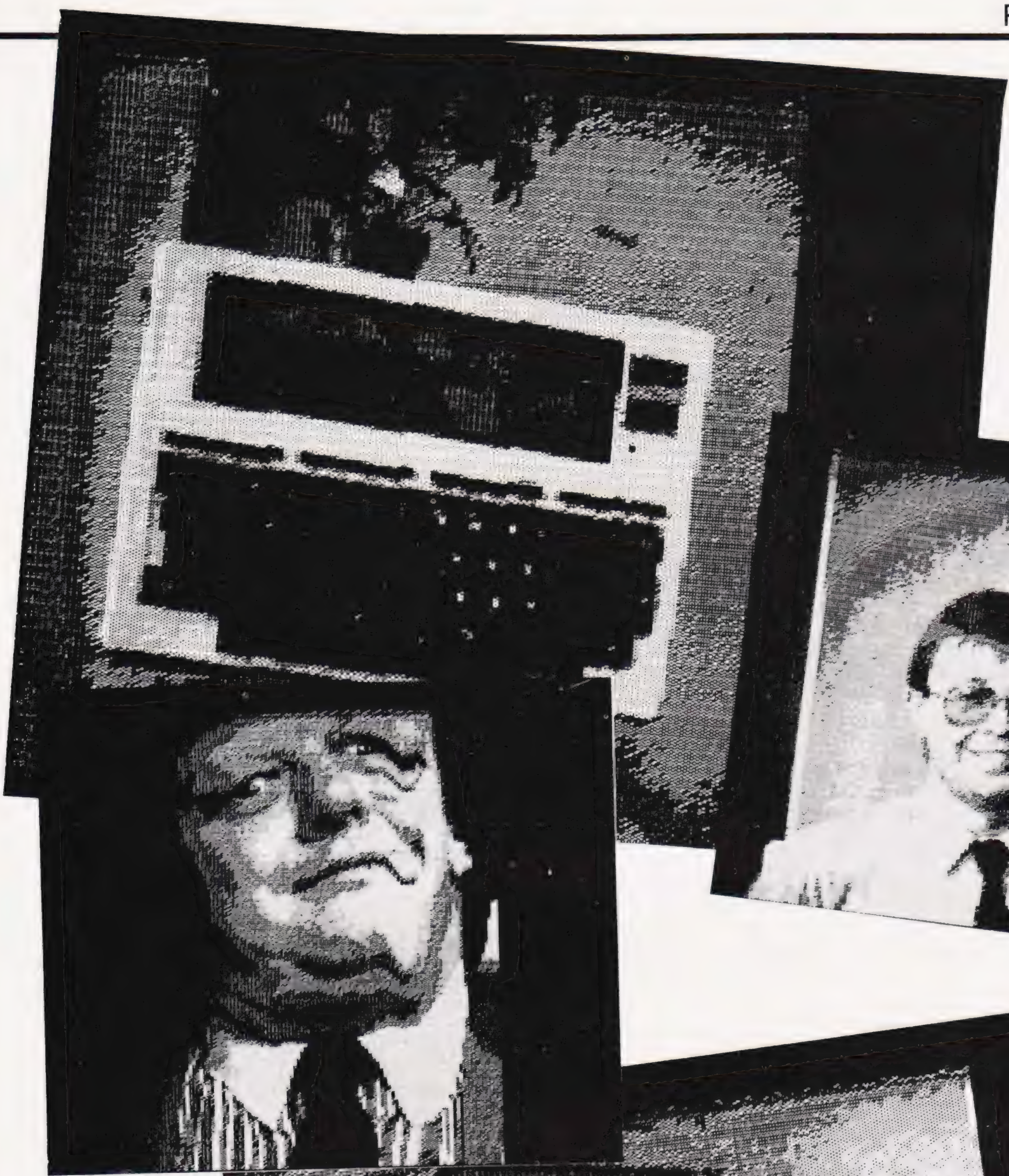
The software supplied with the basic system is contained by three discs. The first of these is the 'photo disc' which enables the user to simply capture an image, display it on a Mode 2 screen, and store it for future use. The 'MicroSight 1' disc is a more flexible beast, in that it will allow an image, once captured in the micro's screen memory, to be manipulated to give, in the case of the 'border' facility, an outline of an image with the main body of the image completely removed. The 'area' facility performs almost the antithesis of this function, in that much of the grey-scaling is removed to leave a crude, but very interesting silhouette-like image consisting only of black and white levels.

The image held in the micro's graphics RAM is refreshed, quite conveniently, using the 'Refresh' option. This option displays a 128 by 128 pixel image situated at the bottom-left of the screen, and is designed to quickly display an image which is the current object of the camera, in order that adjustments to focus and so on can be made without waiting an age for the whole screen to be updated. A 'Threshold' facility allows direct adjustment of the contrast of the image, while the 'Colour' option expands the





Some examples of the images that can be obtained using the MicroSight package, dumped in black-and-white to an Epson printer. Above, you can see how weird effects can be obtained by moving the subject during the scanning of the image.



Left: the amazing two-headed editor! Don't move during exposure, or strange results can occur: see above.

128 by 128 image to 200 by 200 with five scales of grey.

'Dumping' of images — to disc or printer — is also catered for, and while the printer dump routine supplied uses Epson protocols, any screen-dump utility could be used once the required image has been stored on disc. Finally, the screen menus can be switched on and off, leaving just the captured image present upon the screen.

WHAT'S ON THE MENU?

The options are spread over two menus, and although I am ashamed to admit it, this split-level approach had me baffled for a time as some options are accessable from both menus and no explicit indication is given as to which menu is currently on display (simple 'MENU 1' and 'MENU 2' headings would have solved this problem).

The final disc supplied with the basic system — 'Microsight Hi-Res' — offers virtually the same facilities as the Photo disc, but for two important exceptions. While both the Photo disc and the standard MicroSight programs use a Mode 2 screen to display images, the Hi-Res disc uses Mode 0 which has a graphics resolution of 640 by 256, compared with the 160 by 256 resolution of the Mode 2 screen. The higher resolution mode, however, is only capable of displaying two colours, against the eight colours available with Mode 2, so although the image may have an improved picture 'definition', much image information is lost by the screen's inability to display more than two grey-scales (ie black and white).

The other important difference is that the Hi-res program has a 'slogan' option. This permits character-strings to be superimposed upon a given image, anywhere about the screen — useful for adding titles and lables to captured images.

OPTIONAL EXTRAS

The programs described so far are those supplied as part of the basic system. In addition to these, however, there are two other packages available which are designed to greatly enhance the flexibility of the system. The first of these is a colour add-on called 'MicroSight Colour'. This system, through its ingenious use of red, green and blue filters, allows the system to represent colour images using only the existing monochrome camera. This is achieved by mounting a motor beneath the camera (ours was already fitted), to which a disc containing three filters is added.

The motor is energised under software control such that, at any given moment, only one of the red/green/blue filters is in front of the lens. If, for example, the red filter covers the lens, only red light will be admitted into the camera, and hence the software will compose an image, as displayed upon the screen, from red only. If the disc is revolved to position the green filter in front of the lens and another scan performed, the resulting image will consist of green only. This image is then 'superimposed' upon the existing red-only display. The same process is performed a final time using the blue filter to yield a full, three-colour image. The overall effect of the layered display is interesting, in that the three separate colours combine to provide the illusion of a full-colour picture!

PRICES

Basic MicroSight system for the BBC Micro, Commodore 64, and IBM PC, including monochrome video camera, tripod, Microeye interface, cables, MicroSight 1/MicroSight Hi-Res/Photo discs.	£495 plus VAT
Microscale Image Analysis Software	£295 plus VAT
Colour Add-On including motor, filters, and MicroSight colour disc	£99 plus VAT
MicroSight Interface card for the IBM PC (necessary to link the basic MicroSight system to the PC)	£95 plus VAT

The MicroSight system is also available for the Hewlett-Packard HP 9816 and the ACT Sirius microcomputers. For additional information regarding any of the above systems, contact Digithurst Ltd, Leaden Hill, Orwell, Royston, Herts. SG8 5QH (telephone 0223 208926).

MANIPULATION

The second package available as an extra is the 'Microscale' package. This comparitively advanced piece of software probably deserves a few pages of its own! Briefly, though, this package allows selective analysis to be performed upon images by defining 'windows'. A window is used to designate which portion of an image is to be manipulated, and in this way specific areas of a picture can be transformed into 'area' or 'border' representations, or removed altogether. Also, the relative dimensions of specific objects within an image can be determined, and this feature alone could have numerous practical applications such as sorting differently shaped objects.

DOCUMENTATION

How sad it is that poor documentation often spoils what might otherwise be a well-presented system. The manual which accompanied our system was an example of such documentation, containing not only fundamentally inaccurate statements (on the very first page a routine is referred to which does not even exist as part of the system software), but also containing references to non-existent chapters! However, what the manual lacks in presentation, it attempts to make up for in detail. A source listing is given for every routine — a virtually unknown phenomenon — and the operation of each routine is described in such lucid terms that a quick glance through the manual is almost an education in itself! This only goes part of the way towards compensating for its basic inpenetrability, though, and I think I would try to avoid referring to the manual whenever possible.

EASE OF USE

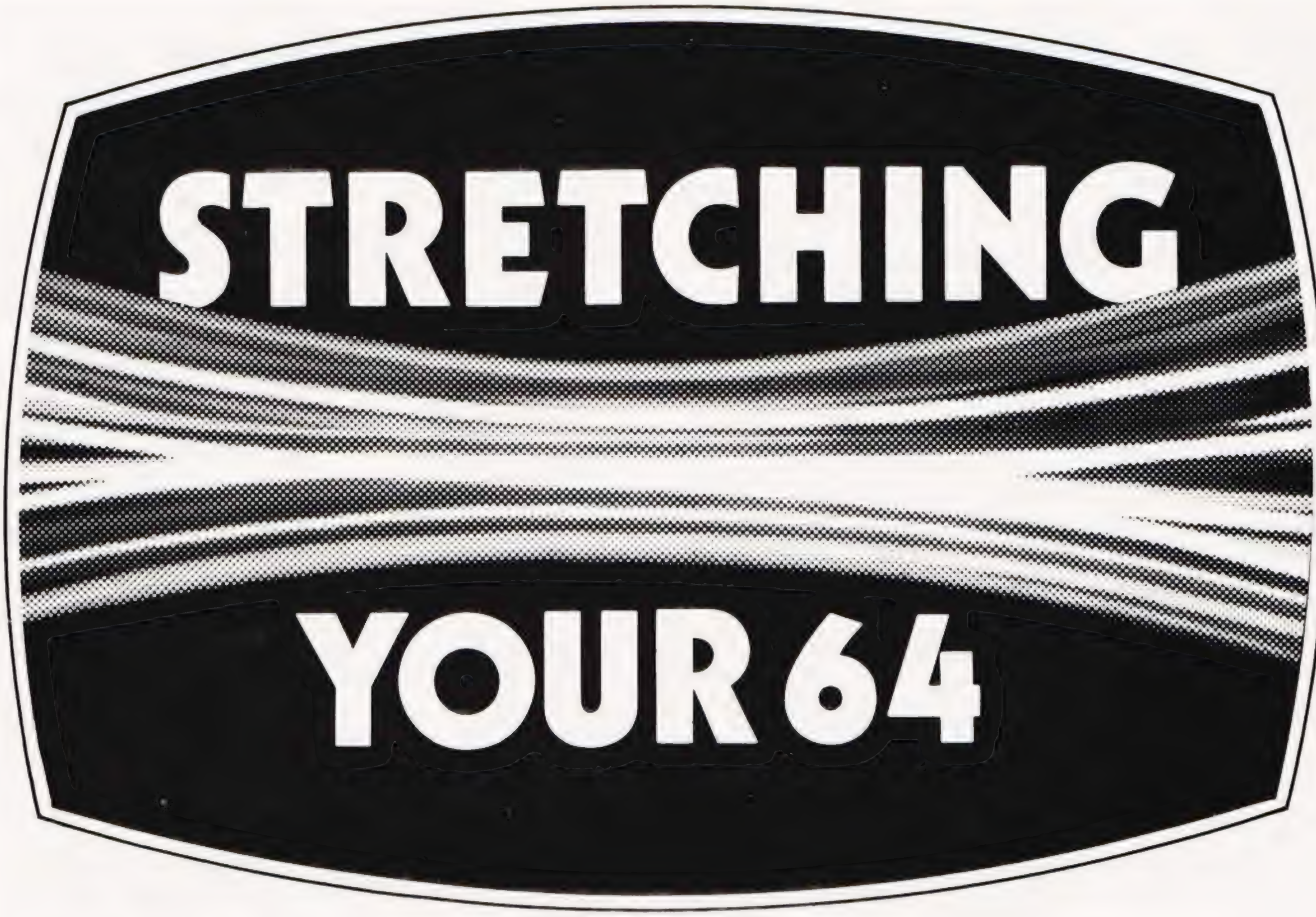
Operating the system was quite straightforward in spite of the manual, and once the basic requirements such as focus, illumination and aperture had been adjusted, obtaining an image was usually a simple matter of pressing the relevant key. Some of the more advanced features were a little difficult to get to grips with, but this was due to unfamiliarity rather than any great failing in the system itself. As mentioned, the camera was sensitive enough to produce good images under poor lighting conditions if used directly with a motor, but to obtain similar results with the MicroSight system as a whole required slightly more favourable — but by no means exceptional — conditions.

The presentation of on-screen information (menus and so on) was quite disappointing, despite the graphics capabilities of the BBC Micro. In fact, the appearance is quite typical of software produced by a hardware company and given the price of the system as a whole, a little more effort should have gone into improving the intelligibility and overall friendliness of on-screen information and prompts. As I have already mentioned, the system had a tendency to leave the user in the dark as to what was happening at any given moment, although this had a beneficial side-effect, in that it encouraged me to take more care when selecting options from the menus than I might otherwise have done!

CONCLUSION

Given the cost of the MicroSight system, those wishing to purchase a relatively inexpensive image analysis system could do far worse than to choose this one. It is very good value for money, and I still find it difficult to accept that MicroSight comes complete with a camera. The software, despite its disappointing appearance, functioned well in use and always — well, usually — gave what was required of it. The documentation was the main failing of the system. The illogical layout of sections within the manuals was barely excused by my amazement at finding complete source code listings for the routines — a rarity indeed!

Competition in this area of the market is not particularly fierce at present, but Digithurst's MicroSight system will, I think, establish itself as *the* low-cost, image analysis system for micros.



To a lot of people, the Commodore 64 is merely a home computer. A handy, entertaining, semi-serious piece of kit which all the family can enjoy. Few people appreciate, let alone exploit, its hidden applications potential. Because, to be perfectly honest, no one has ever bothered to produce any software to significantly boost its performance and exploit all the exciting possibilities.

Well, we're about to change all that. HANDIC have pioneered four programmers aids that really do stretch the imagination and latent potential of your Commodore 64.

Take a look . . .

GRAF 64 With GRAF 64 you can study complicated mathematical functions by their graph. You can also define a function and plot the graph in high resolution within an w-axis range. Graphical areas far from the original can be studied by a blow-up of a specified range. There is also a special routine for computing the integral of a function within a range specified by you. **£29·95**

MON 64 Machine Code Monitor on cartridge — you do not have to wait while the cassette or diskette is being loaded between runs and debugging attempts. The package includes two (Yes 2!) monitors, so that it is possible to decide what parts of the memory NOT to use. It is, so far, the only monitor on the market that works with all 64K of the CBM 64 memory. MON 64 has built-in help screens and a reset function. There is a command that lets you write ASC11-code directly into the memory. Except for the special functions the MON 64 has the following facilities: Assemble, Breakpoints, Disassemble, Go, Hunt, Interpret, Load, Save, Transfer, Relocate, Display Memory, Display Registers, Walk, Quicktrace, Modify Registers etc. and many others. **£39·95**

STAT 64 STAT 64 simplifies your work with statistics and graphic displays. STAT 64 will add 19 commands to your Basic language. For example: bar charts (histograms) horizontally or vertically, plotting with 3871 points (high-res 64000), screen dump even if the screen is in graphic mode (not the high-res mode), statistical commands for mean value, standard deviation, variance and other critical calculations. **£29·95**

REL 64 Turn your COMMODORE 64 into a remote Command and Control Centre. Activate burglar alarms, garage doors, locks, electric radiators, switches, thermostats, transmitters, lamps, valves, remote controls, pumps, telephones, accumulators, electric tools, stop watches, central heating systems, ventilators and airconditioning units, humidifiers, televisions, stereos . . . you name it. **£34·95**

Stretch your performance and exploit the hidden potential of your '64. For more information contact our 'Programmers Aids Centre' on 0344-778800. Or simply fill in and post the coupon to us.

All prices include VAT.
All products supplied as plug in cartridge.

I would like you to send by return post further information on the Handic 64 range

Name .... Title ....

Company ....

Address ....

....

Tel ....

5 Albert Road, Crowthorne
Berkshire, RG11 7LT
Tel: 0344 778800 Telex: 849426

handic
software Ltd
CT 2
Thinking ahead

EASYCODE PART 4

Simon N. Goodwin

And so to the final act, where we introduce a few characters from the real world of microprocessors and compare their instruction sets to that of our BASIC simulation.

In this final instalment of the the 'Easycode' series we'll compare our invented machine-code with the real thing, and explain some of the features of common microprocessors. We've got examples in Easycode, Z80, 6502 and 6809 code, and we'll compare the performance of all of the popular processors.

THE DIFFICULT BIT

When this series began we deliberately skipped over an area which is normally introduced very early in machine-code tutorials — the idea of 'bits'. This isn't a very hard idea to grasp, but it is difficult to see why it is important when you've no background knowledge of machine code. By now you should know enough to be able to see how bits can be useful.

You've almost certainly been told, at one time or another, that computers store information in 'ones and noughts'. In other words, a voltage can be either present or absent at a point inside a computer or memory chip. If the voltage at a point is much over 2V we generally call the level 'Logic 1'. Less than a volt usually corresponds to 'Logic 0'. (Special purpose computers sometimes use different levels: officially, in micro systems, a '0' is represented by 0V and a '1' by 5V).

The important point is that a computer can't store other values. High voltages are '1's, lower voltages are '0's. The exact line between the two levels, or 'states', varies between components but it is generally between 1 and 2V. There is no 'Logic ½' level, and no 'Logic 3' or 'Logic −2'. Computer storage is composed of cells — or 'bits'. Each bit contains either the value '0' or '1'.

BASE 100

Easycode differs from real computers because each location — the smallest unit which can be handled — can contain any value in the range from 0 to 99. If you want to store a larger number in Easycode, you have to use more than one location. To store a value up to 9999 you use two locations, one for the 'units' and another for the 'hundreds'. In principle this could be extended indefinitely — to store the number of people in the UK you'd need four locations, for units, hundreds, tens of thousands, and millions. The limited capacity of an individual location doesn't matter much, so long as you can manipulate groups of them together.

BASE 2

A single bit is the most limiting capacity of all. You can use it to represent the answer to a 'yes/no' question but it isn't useful for much else on its own. Micro systems recognise this fact and most instructions deal with groups of bits rather than individual ones. An 'eight bit' computer is one which uses, most commonly, groups of eight bits, or 'bytes'.

Most popular micros use eight bit processors, although they often have a few facilities for handling information in larger or smaller amounts. There are jargon terms for each amount: 'words' (16 bits) — 'long words' (32 bits), 'pages' (often 256 bytes — 2084 bits), 'segments' (typically 65536 bytes) and lots of others, specific to individual processors. There are units smaller than a byte, too: a 'nibble' is two bits and a 'nybble' (note the 'i' and the 'y') is four bits.

The more bits there are in a unit, the more values can be represented. In a nibble you can store four different values: 00, 01, 10 and 11. This is a notation called 'binary' (meaning twofold) since each bit may be either a '0' or a '1'.

The number of values which can be stored in a certain number of bits is found by multiplying two (the number of possible states) by itself for each available bit. In two bits we can store four values, in four bits we can store 2 x 2 x 2 x 2 = 16 values, and so on. If you're not sure of this, check it out by writing all of the permutations of four bits. Once the rule is known we can say that 256 values will fit in a byte (eight bits), 65536 values will fit into a computer word (16 bits), and so on.

DECIMAL AND BINARY

It is helpful to be able to write numbers in binary, but sometimes they are rather unwieldy. It is much easier to say that the Spectrum's display memory is located between address 16384 and 23295, than to use the binary forms 100000000000000 and 101101011111111, even though the binary does represent the pattern of voltages used inside the computer! We need a consistent way to convert binary into decimal.

If we call the right hand digit the 'units', the next digit the 'twos', then the 'fours', 'eights', 'sixteens' and so on, we can convert from binary to decimal quite easily. The sequence of values ('powers of two' or 'weights'), from the right, goes 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 (a 'K'), 2048, 4096, 8192, 16384, 32768, 65536 and so on. You pick up this sequence quickly as you learn machine code.

TABLE 1

Weight:	128	64	32	16	8	4	2	1	
Binary:	1	0	1	0	1	0	1	0	
Decimal:	128	+	32	+	8	+	2	=	170

Binary to decimal conversion.

As an example, Table 1 converts the number 10101010 from binary to decimal. You should be able to confirm that 11111101 in binary is 253 in decimal, and 01010101 (binary) is 85 (decimal).

SHIFT WORK

There is a hidden advantage of binary arithmetic — it is better suited to electronic addition, multiplication, and so on. Even pocket calculators use binary internally, like computers, and then convert the values as required. You may have spotted the technique used when you tried out the examples above.

A microprocessor is much better at moving bits around than it is at complicated arithmetic operations like multiplication. There are electronic ways of multiplying or dividing in a single step, but they're certainly not trivial! What is easy is moving bits up and down in a register — so-called 'shifts' (if the end values are thrown away) or 'rotates' (if the end values re-appear at the other end).

Look back at the binary forms of 85 and 170. To convert 85 into 170 all you have to do is shift the binary one place to the left. Such a shift is, in effect, a 'multiply by 2' operation. Similarly a shift to the right corresponds to a 'divide by 2'. By convention the bit which 'falls off' after a shift or rotate (the remainder, here) is shunted into the carry flag.

Earlier in this series we performed a multiplication by adding a number to itself repeatedly. We warned that this was not the

technique used in real computers — now we can see the real technique in action!

PROBLEMS MULTIPLYING

It is fairly easy to program a multiplication using a combination of shifts and additions (hardware addition is simpler than multiplication — all microprocessors support it). It becomes even easier once you realise that we're using binary, since the ones and noughts in the binary can be used to indicate when we should shift the result and when we should add the value to be multiplied.

As an example we'll multiply 23 by 11, using binary throughout. 11 decimal (the multiplier) is 1011 in binary, and 23 (the multiplicand, or 'other number!') corresponds to 10111. The operations are 'dry run' in Table 2.

We end up with the binary value 11111101, which — amazingly enough — is 253 in decimal! This may seem like

TABLE 2

STEP	ACTION	MULTIPLIER	RESULT
0	Set multiplier, clear result	1011	00000000
1	Shift result left (!)		00000000
	Rotate multiplier left. If a	0111	
	'1' fell off, add multiplicand		00010111
2	Shift result left		00101110
	Rotate multiplier left. If a	1110	
	'1' fell off, add multiplicand		
3	Shift result left		01011100
	Rotate multiplier left. If a	1101	
	'1' fell off, add multiplicand		01110011
4	Shift result left		11100110
	Rotate multiplier left. If a	1011	
	'1' fell off, add multiplicand		11111101

Binary multiplication.

magic, but it does work reliably for any size of number, so long as you carry out one shift/rotate step for every bit in the multiplier.

If this still seems confusing, try thinking of it as a process of repeated additions with shifts intermingled. An add followed by four shifts corresponds to 16 adds, an add before two shifts corresponds to four adds, and so on. The later a bit appears in the multiplier, the less additions it represents. Each position represents half as many additions as the position to its left.

There is a 'useless' shift in step 1 — this shifts the value zero, making absolutely no difference! It has still been performed to emphasise that the procedure is absolutely identical at each step: shift the result, shift the multiplier and add the multiplicand IF there was a carry from the multiplier). There should be as many steps as there is room for bits in the multiplier. Here we've used four bits, hence four steps are needed to line up the result correctly.

You may need more bits for the result than you did for the original number, just as in human arithmetic. In decimal, 99 times 90 (two-digit originals) gives a four-digit result: 8910, for those with boggling brains or flat calculator batteries! The number of digits (or bits) in the result is never more than the total number of digits in multiplier and multiplicand.

Table 2 presumes that you either know or can work out how to add binary numbers. The process is actually very simple, although computers are best at it so it is OK to skip the arithmetic as soon as you understand how it works! Binary addition is done from the right-hand digit leftwards, just as you were taught (with decimal) at school. A '1' can be 'carried' to the left as usual, so

TABLE 3

```
0 + 0 + No carry = 0
0 + 0 + carry    = 1
0 + 1 + No carry = 1
0 + 1 + carry    = 0, carry 1
1 + 0 + No carry = 1
1 + 0 + carry    = 0, carry 1
1 + 1 + No carry = 0, carry 1
1 + 1 + carry    = 1, carry 1
```

Adding two bits together.

you have to take the previous 'carry' into account for every digit after the first. For each column of binary there are eight possibilities, shown in Table 3.

In principle this rule can be used for addition inside the computer, although tricks are used so that the processor doesn't have to wait for all the rightmost bits to be added before it can work out the result (including carry) for the leftmost. These tricks are interesting, but rather beyond the scope of this article!

ILLOGICAL, CAPTAIN!

There is one more application of bits which it is useful to understand. This is the idea of 'logical operators': not spies from the planet Vulcan, but actions based on simple binary rules. Logical operators allow a programmer to pack different information into any bit or group of bits within a location. Shifts and rotates can be used to line the bits up before they are stored or recalled. Table 4 summarises the effects of the operators NOT, AND, OR and XOR.

The simplest logical operator is called 'NOT', 'inverting' or 'ones-complement'. The effect of the 'NOT' operation is to flip the values of each individual bit. Every one becomes a nought, and vice versa.

The other logical operators involve two numbers — the data, and a so-called 'mask' value which is used to determine the result. All of the operators are commutative, which means that you get the same result whichever value you say is the 'mask' — in normal arithmetic, addition is said to be commutative, since 7 + 2 = 2 + 7, whereas subtraction is not: 7 − 2 is not the same as 2 − 7 (or so my bank manager insists!).

The AND operator produces a result which contains 'ones' at every position where the data AND the mask contained a '1'. This is useful when you want to check some of the bits in a number and ignore the others. To test whether a number is odd or even, for instance, you just AND it with 1. If the result is zero then the number is even, otherwise it contained a 'unit' and must be odd.

The OR operator produces a result which contains '1' at every position where the data OR the mask (or both) held a '1'. This is a useful way to set certain bits in a register. OR A;4 will set the bit third from the right (the 'fours' column) in register A, whether or not it was set before.

The last operator is the most devious. The 'exclusive OR' (XOR or EOR) operation sets a bit in the result whenever ONLY ONE of the corresponding bits in data and mask is set. XOR A;1 would

TABLE 4

```
Logical NOT:
A          10011101
NOT A      01100010

Logical AND:
A          10011101
B          11110000
A AND B    10010000

Logical OR:
A          10011101
B          11110000
A OR B     11111101

Logical XOR:
A          10011101
B          11110000
A XOR B    01101101
```

Logical operators.

make the contents of A odd if they were previously even, or vice versa. If A is an eight-bit register, XOR A;255 will produce the same result as NOT A — all of the bits in A will be flipped (since, for any bit, '1' XOR '1' is '0' and '0' XOR '1' is '1').

The key thing to remember is that if you XOR a location with the same mask twice, you get the first number back. This is very useful for applications like graphics, when you want to store and then erase a pattern.

REAL MACHINE-CODES

This series ends with a look at the three most common forms of 'hobby' machine-code — Z80, 6502 and 6809 code. We'll also mention the 6800, 8080, 8085, 8086, 8088 and 68008 processors in passing!

There are, broadly speaking, two different types of eight bit microprocessor — Motorola-style and Intel-style. Motorola and Intel are both American chip-makers. Firms such as RCA, Texas and National Semiconductor have produced different designs, but they haven't caught on with hobby machine builders. Figure 1 shows a family tree of the most popular processors.

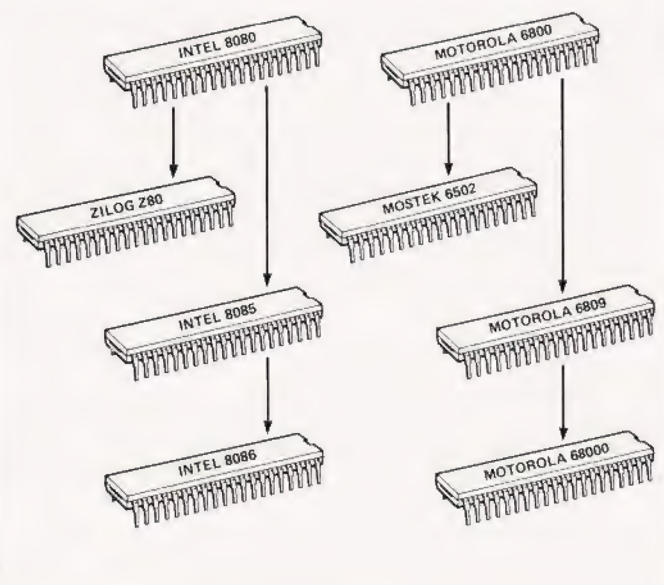


Fig. 1. The microprocessor family tree.

The first 'hit' micro was the Intel 8080, a development of an earlier calculator chip called the 8008. Later in 1975 Motorola produced the 6800 chip, a competing processor with a totally different internal design.

INTEL

Processors which follow the Intel design tend to have lots of registers and instructions, all of which have different, specialised applications.

The speed of a processor is determined by the timing clock signal it receives. Intel-style machines tend to take a number of pulses of the clock before they produce results (usually between four and 20 pulses, depending upon the instruction).

Common Intel-style processors are the 8080, 8085, 8086, 8088 and Z80. The 8085 and Z80 were both directly modelled on the 8080, to the extent that they share much of their instruction-set.

MOTOROLA

Motorola-style processors have relatively few registers and instructions. Instructions tend to be simple and consistent between registers, so that each register can be used in much the same way as the others (this is less true of the 6502).

Motorola established the idea of using 'instruction pre-fetch', which means that one part of the processor gets on with fetching information from memory while the other part decodes and performs the instructions. This 'parallel' principle is extended elsewhere in the processor, so that instructions take few clock pulses to be performed — generally between one and six pulses. This is one of the reasons why the Motorola-style BBC Micro (6502 at 2 million clock pulses a second, or 2 MHz) can out-perform the Lynx or Spectrum with their Intel-style Z80s running at twice the clock speed.

The Motorola 68008 processor used in Sinclair's QL computer has been criticised because it is a 'cut-down' version which addresses memory in bytes rather than 16-bit words. In fact the loss in performance is quite small — generally only about 30 per cent — because of the way the processor can access memory at the same time as internal operations are performed.

By way of contrast, many business machines use the Intel 8088 microprocessor, a cut-down model of the 16-bit 8086. The 8086 does not 'look ahead' in the same way, so the performance of the 8088 is degraded much more.

BATTLE OF THE BYTES

Listing 1 shows an Easycode program similar to one introduced earlier in this series. The program adds up a list of 10 numbers and stores the total in memory. Listings 2, 3 and 4 contain equivalent programs for the Z80, the 6502 and the 6809. The overall structure of the solution is the same in each case, although the detail has been altered. The programs are not claimed to be the best possible solution for each processor, but they do show the differences between the machines quite accurately.

Each real program starts with an ORG statement which tells the assembler where the code and data are to be stored. The value you pick depends upon the use of memory in your computer — it doesn't really matter where you pick so long as the memory is not already in use.

Words like DEFW, DW, and FDB are used to initialise a word location to a given value (zero, in the listings). Similarly DEFS, DS and RMB reserve a specified number of bytes for data (without explicitly storing a value). These words (called 'pseudo-ops' since they don't cause any processor operations) vary between processors and assemblers. The examples above were tested using EDAS on a Video Genie, AMAC on an Atari 800 and DASM on a Dragon 32.

Each assembler (other than Easycode) allows lines to be named or 'labelled'. This allows you to refer to data and code by name rather than address. On the Z80 and 6502 you define labels by following them with a colon — on the 6809 you prefix label names with an '@' sign.

```
 0:  11  81  LOAD X;81        Point to start of table+1
 2:   1   0  LOAD A;0         Fetch a zero to...
 4:   3  70  STORE A;@70      Clear the hundreds count
 6:   2  80  LOAD A;@80       Fetch first value
 8:  19      ADD A;@X         Add the next value
 9:   8  19  JUMPNC;19        Don't carry unless we must
11:  20      PUSH A           Save the units count
12:   2  70  LOAD A;@70       Fetch the hundreds
14:   5   1  ADD A;1          Count one more hundred
16:   3  70  STORE A;@70      Store the result
18:  22      POP A            Restore the units
19:  16  89  SUB X;89         Has X reached 89?
21:   9  26  JUMPNZ;26        Jump on if X was not 89
23:   3  71  STORE A;@71      We've finished, store units
25:  25      RETURN           Go back where you came from
26:  15  90  ADD X;90         Advance X to the next value
28:  10   8  JUMP;8           Go and add the next value
```

Listing 1. The Easycode adder (30 locations of code). It adds the values in 80-89, storing the result in 70 and 71.

THE Z80 PROGRAM

The Z80 processor is very popular with hobbyists, mainly because it has lots of 'special case' instructions. A very clever programmer can consequently produce very clever programs! However, it does take a while to learn the idiosyncracies of the Z80.

In Z80 assembler a number or register-name in brackets represents a pointer to a location, so tht LD A, (100) would LoaD

```
        ORG     50000       ;Start assembly at address 50000
TOTAL:  DEFW    0           ;Room for the result (word value)
TABLE:  DEFS    10          ;Space for 10 data bytes

ADDUP:  LD      B,9         ;The number of 'adds' required
        LD      HL,TABLE    ;Registers H & L point at TABLE
        LD      C,0         ;C counts multiples of 256
        LD      A,(HL)      ;Fetch the first value into A
NEXT:   INC     HL          ;Point to the next value
        ADD     A,(HL)      ;A = A + what HL points at
        JR      NC,NOCAR    ;Rush on unless A overflowed
        INC     C           ;Add 1 to (increment) register C
NOCAR:  DJNZ    NEXT        ;Decrease B (B=B-1) Jump if not 0
        LD      L,A         ;Copy 'low' byte into register L
        LD      H,C         ;Copy 'high' byte to register H
        LD      (TOTAL),HL;Store result (H and L) in TOTAL
        RET                 ;Go West, young processor!
        END     ADDUP       ;This marks the start and end
```

Listing 2. The Z80 adder (22 locations of code). This adds the values in TABLE, storing the result in TOTAL.

the contents of location 100 into register A (oddly, Z80 instructions move data from right to left!) and LD A,100 would put the value 100 into A. Some of the eight-bit registers can be used in pairs as 16-bit pointers: the H and L registers are used in this way in Listing 2. The DJNZ instruction is a typical compound Intel instruction. It combines a count-down (in the B register) with a conditional jump if the result is not zero.

THE 6502 PROGRAM

The 6502 is the most consistently-eight-bit processor, so Listing 3 has to store the 16-bit result in two eight-bit stages. Even the stack pointer is an eight-bit register, so a maximum of 256 bytes can be PUSHed. By way of compensation the 6502 has fast, useful

```
        ORG     1540      ;Start assembly at address 1540
TOTAL:  DW      0         ;Room for the result (word value)
TABLE:  DS      10        ;Space for 10 data values

ADDUP:  LDY     #1        ;Y selects each value in turn
        LDX     #0        ;X counts multiplies of 256
        LDA     TABLE     ;Put item at start of TABLE in A
NEXT:   ADC     TABLE,Y   ;Add value at TABLE+contents of Y
        BCC     NOCAR.    ;Rush on unless A overflowed
        INX               ;Add 1 to (increment) register X
NOCAR:  INY               ;Point Y at the next value
        CPY     #10       ;Compare Y with 10
        BNE     NEXT      ;Branch (jump) to NEXT if Y<>10
        STX     TOTAL+1   ;Store X in high byte of TOTAL
        STA     TOTAL     ;Store A in low byte of TOTAL
        RTS               ;We've finished (horray)
        END     ADDUP     ;This marks the start and end
```

Listing 3. The 6502 adder (25 locations of code).

instructions for 'indexed addressing', such as ADC TABLE,Y which takes the address TABLE, adds the eight-bit value in register Y, fetches a byte from the address totalled, and adds that byte to the value in the accumulator! In effect the instruction combines an adjustable array acess and an add in one step.

Motorola don't use brackets in the same way as the Z80 assembler. To load the VALUE 100 into register A you mark the number with a hash: LDA #100. To fetch the byte at address 100, use LDA 100.

Motorola-style computers call conditional jumps 'branches' to show that they use a trick called 'relative addressing'. The instruction doesn't contain the target address of the jump — instead it stores a positive or negative 'offset' between the following instruction and the one which might be fetched instead. Consequently BCC NOCAR is stored as 144 (the code for BCC) followed by '1' to indicate that one byte should be skipped (the INX) if the carry flag is clear. Relative branches make programs shorter and easier to move about in memory.

THE 6809 PROGRAM

It is a shame that the only common machines using the 6809 are the Dragon and Tandy Colour computers, which have weaknesses elsewhere in their design. The 6809 has lots of 16-bit instructions, an extra stack (so you don't need to get data and CALLs muddled up), even more ways of addressing data than the 6502, and a one-step built-in multiply operation. Despite its comprehensiveness, the instruction-set of the 6809 is the most consistent of any eight-bit processor, making it easy to learn and use efficiently. A seasoned CT contributor, Mike James, has written **The 6809 Companion** (Babani), which is an excellent cheap reference if you'd like to find out more for just £1.95.

```
        ORG     20480     ;Start assembly at address 20480
@TOTAL  FDB     0         ;Room for the result (word value)
@TABLE  RMB     10        ;Space for 10 data values

@ADDUP  LDA     #10       ;The A register counts the values
        LDX     #0        ;Clear result, which will be in X
        LDY     #@TABLE   ;Point register Y at the TABLE
@LOOP   LDB     0,Y+      ;Copy data at Y to B, add 1 to Y
        ABX               ;Add register B to total in X
        DECA              ;Count one less value to be added
        BNE     @LOOP     ;If A is not zero, go to LOOP
        STX     @TOTAL    ;Save the result at TOTAL
        RTS               ;Easy, eh?
        END     @ADDUP    ;This marks the start and end
```

Listing 4. The 6809 adder (19 locations of code).

Turning to our example program, four registers are used — the 16-bit index Y to point to the data, index X to store the total so far, A to count data items and B as a temporary store for each item.

"LDB 0,Y+" adds 0 to the contents of register Y (the zero is a dummy value in this case) and puts the contents of the total address into register B. The '+' tells the 6809 to add one to register Y (selecting the next byte in the table) while it is copying the byte into B! "ABX" is a rare but useful instruction which adds the 8-bit number in B to the 16-bit total in X.

CONCLUSION

With a little luck this series has explained the essence of machine code programming. If some of the details are a little unclear, don't worry — you learn programming by doing, not by reading. This series really set out just to encourage you to get started!

The next step is to buy, borrow or steal an assembler (make sure you get all of the instructions!) and buy a good book on your chosen processor (bad luck if you've got a TI 99/4A). The **Programming the 6502/6809/Z80** series by Rodnay Zaks (Sybex) will stand you in good stead, so long as you are careful not to buy first edition copies, which tend to be plastered with errors.

We look forward to seeing your concise machine-code masterpieces in CT soon!

BEEBUG'S TOOLKIT

Peter Freebrey

When you're programming on the BBC Micro and there's a spanner in the works, you'll find yourself wishing for a Toolkit to help you out. We look at a comprehensive ROM from Beebugsoft.

What is a toolkit? Those that might have been unwary enough to have read any of my articles in the past, will perhaps have realised that I subscribe to a variation of a maxim that I was taught at school (yes, I can remember that far back!). The maxim? It was hammered into our young heads by a Mathematics teacher — *the best mathematicians are lazy.* The inference was that we should not slog away with pages of mathematical manoeuvring when a little thought would present us with a short elegant solution.

I am not highly critical of inelegant or unstructured programs — I feel that if a program works to *your* satisfaction, let other programmers argue about improving it! But I *am* lazy: I write quite long programs and any aid that can help me concentrate on the program and free me from a lot of hard work and key pushing is a must. This is the purpose of a toolkit: it supplies the user with a number (ideally all!) of those commands/routines that you wish the computer manufacturer had supplied in the first place. What these routines are will become apparent as we proceed. Let us first look at what the Beebugsoft Toolkit is and how to get it operational.

The Toolkit comes as a ROM chip, a 28-pin integrated circuit chip that *you* are going to have to install in your computer. After I don't know how many years involved with electronics, I still do not relish delving inside a piece of equipment with which I am unfamiliar. Have no fear, though; Beebugsoft supply some of the best step-by-step instructions I've met yet. You would have to be very fumble-fingered to not find it as easy to install as I did. Everything is clearly explained and even the small point that some machines have the fixing screws labelled and some do not is covered.

TURN ON

Having installed your Toolkit and typed ★HELP and RETURN, you should now be looking at something like:

TOOLKIT 1.10
TOOLKIT
INFO

Type ★HELP TOOLKIT and RETURN: you will be presented with a list of the commands available. Type ★HELP INFO and RETURN and this will be replaced with a list containing various status information:

- the current values of FX3 to FX12 inclusive.
- the current values of LISTO, WIDTH and @%.
- ERR, ERL and the last REPORT error message.
- current memory information, bytes free, program size, PAGE, TOP, LOMEM, HIMEM and the next free location.

At least you now know that the chip you've just fitted is doing something. All the commands may be called from within a program, although the results may not always be terribly useful — would you really want to RENUMBER the program lines in the middle of a working program?

The 32-page instruction booklet that comes with the Toolkit is helpful and clear, with all the commands explained in a very straightforward fashion. Toolkit commands when entered are preceded by an '★' in the form ★RENUMBER, ★CHECK or in shortened forms followed by a full-stop. To eliminate possible conflict with any alternative ROM commands that may use the same keyword as Toolkit, all Toolkit commands may also be entered prefixed by the letter 'B' such as ★BRENUMBER, ★BCHECK and so on.

BY YOUR COMMAND

In all there are some 25 commands, in alphabetical order these are:

★CHECK Will verify the contents of a file from disc or cassette. It performs a byte-by-byte comparison of the memory contents with the contents of a specified file. A BASIC program may be CHECKed simply by ★CHECK "program name"; alternatively start and end addresses may be specified and the relevant memory locations CHECKed against a file, so ★CHECK "file name", 1900,1CE8 would verify the thousand bytes between memory locations &1900 and &1CE8.

★CLEAR Will clear all variable assignments much the same as the BASIC statement CLEAR, but ★CLEAR will also set each of the Resident Integer Variables (A% to Z%) to zero. The Resident Integer Variables are special in that they are permanently allocated space in memory, and once assigned they will not be initialised or destroyed by RUN or NEW as are all other variables. Some care must be exercised when using ★CLEAR as some programs use the permanence of the '%' variables as links between programs.

★EDIT Enables the user to enter the Screen Editor either from the start of the program —

★EDIT — or at a specified line number — ★EDIT 100. In the EDIT mode you may edit a line without the need to COPY the whole line (thank God! — Ed): characters may be inserted or deleted and the cursor keys either move the cursor left and right along the program line or to the next program line above *or* below. With the addition of the SHIFT key you may move to the beginning or end of a line or to the first or last program line. An extra line may be inserted by the use of CTRL and TAB. Any alterations made are *automatically entered* on moving to another line. When using 'up' cursor the preceding line is displayed (the cursor does not *scroll* up): this is initially somewhat confusing as one is used to reading programs down the screen in ascending line order. Use quickly brings familiarity with this system and screen editing is certainly quicker, if slightly more prone to 'user error' than the BBC's usual COPY.

★FREE Repeats the information given when you keyed ★HELP INFO, ie the number of bytes of free memory, the size of the current program, the next free memory location (after the variable storage space) and the values of PAGE, TOP, LOMEM and HIMEM. Not exactly an awe-inspiring command, but like others in Toolkit, because it is there, you will find yourself using it more and more often, especially if you are in the habit of writing long programs!

```
>*HELP INFO

TOOLKIT 1.10
  FX  3 0            FX  4 0
  FX  5 1            FX  6 10
  FX  7 100          FX  8 100
  FX  9 25           FX 10 25
  FX 11 33           FX 12 8

  @%=&0000090A

  LISTO   0          WIDTH  0
  ERR     204        ERL    0

REPORT: Bad filename

Free memory =24826 bytes
Program size=   518 bytes
Next free location=&1806
PAGE=&1900    LOMEM=&1806
TOP =&1806    HIMEM=&7C00

OS 1.20
>_
```

Fig. 1 Typical display using ★HELP INFO.

```
>*MEMORY
1900    0D 00 01 2F 20 F4 20 2A  .../ . *
1908    2A 20 46 55 4E 43 54 49  * FUNCTI
1910    4F 4E 20 4B 45 59 20 41  ON KEY A
1918    53 53 49 47 4E 4D 45 4E  SSIGNMEN
1920    54 2E 2E 2E 2E 2E 2E 2E  T.......
1928    2E 2E 22 4B 45 59 22 0D  .."KEY".
1930    00 0A 1E 20 2A 4B 45 59  ... *KEY
1938    20 30 20 2A 46 58 31 31   0 *FX11
1940    2C 33 33 7C 4D 3A 4D 4F  ,33||M:MO
1948    44 45 37 7C 4D 0D 00 0B  DE7||M...
1950    12 20 2A 4B 45 59 20 31  . *KEY 1
1958    20 52 55 4E 7C 4D 20 0D   RUN||M .
1960    00 0C 10 20 2A 4B 45 59  ... *KEY
1968    20 32 20 54 41 42 28 0D   2 TAB(.
1970    00 0D 11 20 2A 4B 45 59  ... *KEY
1978    20 33 20 47 4F 53 55 42   3 GOSUB
1980    0D 00 10 26 20 2A 4B 45  ...& *KE
1988    59 20 36 20 2A 46 58 35  Y 6 *FX5
1990    2C 31 7C 4D 2A 46 58 36  ,1||M*FX6
1998    2C 30 7C 4D 56 44 55 32  ,0||MVDU2
19A0    7C 4D 4C 49 53 54 0D 00  ||MLIST..
19A8    11 15 20 2A 4B 45 59 20  .. *KEY
19B0    37 20 2A 52 45 4E 55 4D  7 *RENUM
19B8    42 45 52 0D 00 12 16 20  BER.... _
```

Fig. 2 Using ★MEMORY to obtain hex and ASCII memory dumps.

★MEMORY Will give a hexadecimal dump of the memory contents with the ASCII equivalent characters alongside to the right. Default start of the hex dump will be from PAGE but any start/finish parameters between 0000 and FFFF may be specified. Both parameters must be in hexadecimal.

★MERGE provides a *very* useful command not found in BBC BASIC. It will merge a BASIC program file (from tape or disc) with a BASIC program already existing in memory. If both programs have a line with the same number (shame on you!), then the line from the file will overwrite the line currently in memory. A warning about the overwritten line will be displayed on the screen.

★MOVE Will relocate the current BASIC program elsewhere within memory, changes PAGE to the new start address and alters all the necessary internal pointers so that the program will RUN at its new location. Although various applications may be found for ★MOVE it is most useful for moving a program down in memory after LOADing from disc, if there is not enough memory for the program to RUN at &1900!

★NEW This is similar to the BBC BASIC's NEW, except that unlike the latter, ★NEW may be used within a program.

★OLD As with ★NEW, this has the same effect as BASIC's OLD, but again it may be accessed from within a program.

★ON Enables Toolkit Error Reporting, and if you are in Mode 7 various messages will appear in colour. After ★ON any error occurring while a program is running will cause the usual error message, but now the Screen Editor (see ★EDIT above) is entered and editing of the current line is enabled. When this occurs the cursor will be in the approximate position of the error.

On certain occasions BASIC will have read past the error before the Toolkit error handling routine is called, so Toolkit has no way of knowing the

exact position of the error. Nevertheless this routine can considerably assist in error correction, but until you are used to the Screen Editor beware of inserting something by mistake. It is easily done, especially as you don't have to key return to enter the line. Also remember that an early line of your program might read ON ERROR GOTO 20000, which may well invalidate the use of ★ON.

★ONF Similar to ★ON but also sets two of the function keys F0 and F1, where F0 gives ★BEDIT and F1 gives ★BUTIL.

★OFF Cancels the use of ★ON.

★PACK Will compact a BASIC program by removing all unnecessary spaces and/or REM statements (and Assembly comments). After a ★PACK command a message is displayed giving the length of the program after compacting and also the number of bytes saved. Care must be taken on subsequent editing as COPYing a line will not always result in the correct tokens for keywords being recognised — but ★EDIT will! The moral here is really to save ★PACK for the final, final working program.

★RECOVER When a 'Bad program' error message occurs it signifies that a program has in some way been corrupted. Often only a small part of the program may be at fault and may be corrected if only the program could be LISTed and edited. ★RECOVER will enable this to be done in many cases. The Toolkit manual gives several ways in which ★RECOVER may be used to regain control after such an occurrence. Although we all know that we should make frequent back-ups of our programs as we develop them, it is good to know that ★RECOVER is ready and willing to help *in extremis.*

★RENUMBER This is an enhanced version of BBC BASIC's RENUMBER and enables the user to renumber any section of a program. Four parameters may be given, allowing the specification of the new start line, increment, old start and old finish. This is a fairly versatile command; naturally all GOTOs and GOSUBs etc are altered throughout, and if a mistake is made, error messages are displayed and the command is not carried out. Like most RENUMBER commands presently available, ★RENUMBER will *not* move a block of lines outside the bounds of the current line immediately below or above the block. This is only a slight limitation to its use but does leave room for possible improvement in the future.

Fig. 3 ★UTIL has been entered and the 'search and replace' function used.

★REPORT Will display the last error message no matter where the error occurred and is equivalent to BASIC's REPORT:PRINT " AT LINE ";ERL. As with other Toolkit commands ★REPORT may be used within a program, for example:

```
10 IF ERR=17 THEN RUN ELSE
*REPORT : REM ** ERR 17 is
Escape
```

★SCREEN Will SAVE the current screen memory to a named file, on cassette or disc, and will function in any Mode.

★UTIL Finally we come to a package of eight utility commands called by the single command ★UTIL. You are then presented with a menu from which to choose the utility required. Individual utilities may be called without invoking the menu by ★UTIL 2 and so on.

(1) **String search** All occurrences of a specified string in the current BASIC program will be found. The line(s) where it occurs will be displayed and the search string displayed in magenta. The search is for the literal string specified, so if a BASIC keyword (GOSUB, REM etc) is to be found, the 'search string' must be prefixed with a '£' to indicate to Toolkit that a tokenised keyword is being sought. There is a 'wildcard' facility where @ may be used as the 'joker' — BACK@UP will find BACK-UP as well as BACK UP.

(2) **Search and replace** This is similar to 'string search' but two inputs will be requested; first the 'search string', followed by the 'replacement string'. Again all lines affected will be displayed with the 'replacement string' highlighted in magenta.

(3) **Move lines** Enables moving a section of a BASIC program to a position elsewhere within the program. This means that the program will have line numbers out of sequence (well, that was what you wanted!) and must be *completely* renumbered. BASIC's RENUMBER cannot cope with this and will fail when it meets the moved block of lines but, not surprisingly, ★RENUMBER takes this in its stride. There are two points to watch out for:

- you have to renumber the entire program, so any structured blocks of routines (DIMs starting at line 100, DATA starting at line 20000 and so on) will have to be renumbered block by block to regain a similar structure.
- GOTOs and GOSUBs etc within the moved block may not be renumbered correctly and will have to be checked individually.

(4) **List procedures and functions** Will give a complete list of all program lines having 'DEFPROC' and 'DEFFN' statements found in the current BASIC program.

(5) **List A% to Z%** Displays current values of all the Resident Integer Variables, both in hexadecimal and decimal notation.

(6) **List numeric variables** Displays all currently assigned numeric variables (other than @% to Z%), again in both hex and decimal notation.

(7) **List string variables** Displays all currently assigned string variables and their present content.

(8) **List arrays** Displays all currently assigned arrays with their dimensions, but the contents of each element are *not* listed.

Note that *currently assigned values* are displayed when selecting 6,7 and 8 above. This means that a program must be RUN, and no lines added or edited before any of these three commands are called, for there to be any values to display!

(9) **Change edit range** Will select that part of the program to be scanned when option 1 or 2 are chosen, so that 'search and replace' may be chosen to be carried out only between specified line numbers.

CONCLUSION

The Beebugsoft Toolkit costs £27 and in my opinion is worth every penny. Since it has been installed in my BBC it has been used extensively and I can find no fault with it. Highly recommended to lazy programmers!

HARD AND FAST

Chris Bidmead

The idea of storing data on a round, rigid magnetic disc has been with us for nearly a quarter of a century. But it is only in recent years that technology has made the hard disc a reality. Here's everything you need to know.

Switch off a microcomputer based on conventional electronics and whatever is in the central memory is lost forever. This part of the computer, sometimes known as the 'core' for historical reasons, is also limited in size, so it needs to be supplemented with backing memory that is large and non-volatile to provide permanent storage of information between sessions.

On cheap home computers and the newer lap-portables, backing storage is provided by audio cassettes. Advantages: supreme cheapness and the general availability of hardware and consumables. Disadvantages: cassettes are slow and tend to be unreliable.

A gramophone record allows you to drop the pickup on any track by sight; with a tape you have to wind through sequentially to get to the next tune you want to play. The idea of using a removable disc with just this kind of 'random access' facility for recording and reproducing digital data was rejected by IBM in 1960, and mainframe technology grew up through the sixties using high-speed digital tape recorders as backing store.

DISC ARRIVE

But the appeal of random access was not lost on mainframe manufacturers, and towards the end of the 60s they developed large magnetic disc subsystems consisting of a number of flat circular plates (called platters) coated on both sides with a magnetic substance. A set of read/write heads rather like multiple gramophone pickup arms interleaved with the platters, but withdrew on power-down so that the whole stack (called a disc pack) could be detached and replaced.

Unlike a gramophone record there was no groove. The data was recorded on concentric rings of the medium called tracks, each track being divided into sectors that could be treated by the system as so many snippets of magnetic tape. Normally there would be no physical delineation of these tracks and sectors: the system would create the tracks by moving the heads in pre-defined steps, and divide each track into sectors by a preliminary writing process called 'formatting' that carved them out magnetically.

Rodime was the first disc drive manufactuer to launch a 3.5" Winchester disc drive in March of this year, and the drive is helping to establish new standard in physical dimensions, disc diameter, interface details and power requirements.

Apart from the cost, one great disadvantage of this kind of storage was that dust particles between the head and the magnetic surface were likely to send the head ploughing into the platter. When this happened the head, the platter and the data on it would have to be replaced, so devices of this type had to be operated in scrupulously monitored air-conditioning.

SOFT v HARD

The technology developed in two directions. By settling for slower access speed and fitting less data onto the medium it was possible to produce small units that used single, cheap replaceable platters that needed no air conditioning. Because they did not have to meet the (literally) rigid requirements of multi-platter disc packs, these replaceable discs of coated plastic protected by a fibre jacket came to be known as floppies. Floppies began to appear as a method of transferring data between IBM mainframes in the early 70s, and the IBM single-sided, single-density 8" floppy is still the only universal standard for data exchange.

An alternative development was to tighten up the specifications, pack more data onto the medium, make all the components smaller and — the final elegant touch — build the air conditioning into the device. Once again it was IBM who pioneered this technology, to give us the Winchester hard disc.

Winchester discs were originally built around 14", and later 8" platters, the same size as the first floppies. As floppies shrank to 5¼" and then to something less than 4", Winchesters have followed suit. Eight inch Winchesters are now a rarity, the 5¼" format, often known as mini-winnies, now being the standard on business micros costing between £2,000 and £10,000. Micro-winnies of the sub-4" format, physically a quarter of the size of mini-winnies, are already beginning to appear, mostly on portable computers.

The first mini-Winchesters to arrive in the late 70s were notoriously unreliable. In part this was because novel hardware technology was trying to squeeze a quart into a pint pot, but in many implementations the systems software writers had overlooked the need to include routines for locking out bad sectors of the magnetic surface discovered during the process of formatting.

Modern 5¼" Winchesters, as long as OEMs have provided them with adequate power supplies, are very reliable indeed. One much-used measure of reliability is the bit-error rate (BER), a function of the average number of bits that can be transferred across the drive/computer interface before an error is made. Today's typical mini-winnie has a BER of one per million, so low that it would take about 70 days of continuous running to test for it using straightforward methods.

Winchester technology improves on floppies much as floppies improve on cassettes, giving greater reliability, speed of access and capacity. Fast, large and cheap backing storage opens up some very interesting possibilities for the modern business user. Unlike the so-called '16-bit Revolution' (see the box) these benefits are real and immediate, and work with existing software.

OVER-DRIVE

Apart from the simple fact that programs load quicker, and data can be processed from disc files faster, the advent of Winchester discs has given a tremendous boost to the huge category of software that uses overlays.

Overlaying is a technique for running large programs in a small amount of core memory by retaining on disc sections of code not immediately needed. Areas of core memory are reserved as a temporary parking lot for this transient traffic, and in the extreme case all that needs to be kept in core for the whole run of the program is the routine for managing the overlays. As far as the user is concerned all the routines in the program are available: if the overlay manager detects the user calling a routine not actually in core, it will organise its transfer from the disc, in the process if necessary destroying (overlaying) a previously-used routine to make room.

Theoretically all this is 'trans-

THE REAL REVOLUTION

Arguably the real revolution in microcomputing has not been the heavily marketed 16-bit processors, but the development of mass-produced mini-Winchesters. The new chips are faster at *processing* (if they're true 16-bit devices talking to a 16-bit bus, which most of them aren't), but most holdups in the micro are due to disc access.

Sixteen-bit chips are also able to address far larger blocks of core memory. But core memory is still expensive, and few business systems actually run to the theoretical 1 Megabyte of RAM permitted by 16-bit processors like the 8086. In any case very little software actually makes use of it.

This Memorex Winchester uses dual platters to give 11.7 Mbytes of store.

parent to the user', a metaphor meaning that the user is supposed to have no need to worry about which sections of code are in core at any one time and which are on disc. But with relatively slow disc devices like mini-floppies, the 'transparency' of waiting for the overlay manager to reorganise the memory begins to be fairly opalescent, if not at times downright opaque. This is what the notorious 'DISK WAIT' message in WordStar is all about.

Speedy backing storage improves this transparency to the point where it really is invisible to the user, lending wings to tired old software and making it possible to run sophisticated applications packages on micros with modest processors and limited core memory.

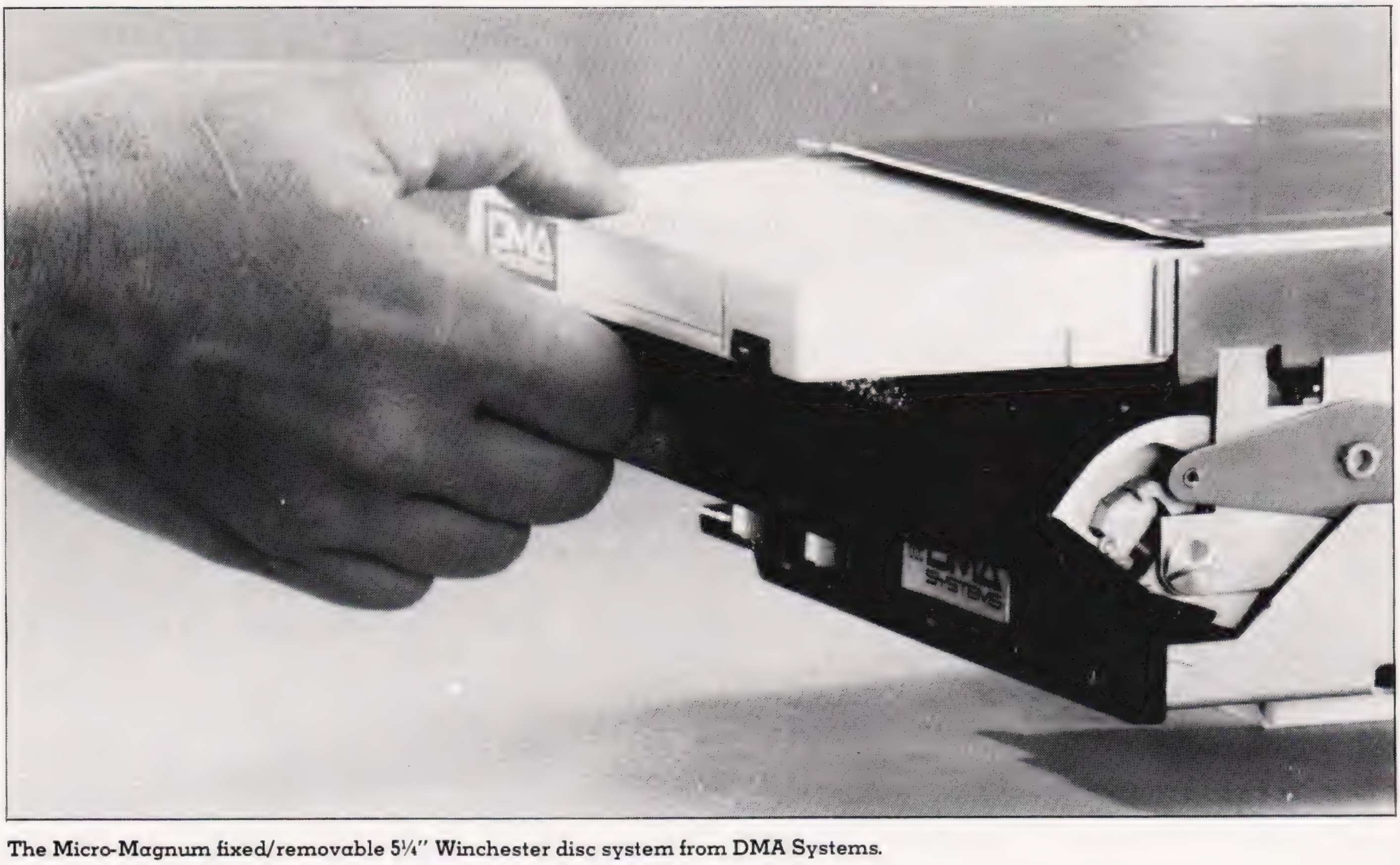
SPACE TO FILL

Assuming that a good typist can enter 1,000 words an hour and spends five hours a day at the terminal (the unions recommend four), a day's work will amount to a little under 30,000 keystrokes. Each keystroke is stored as a single byte, so we won't be far out in calling a day's typing 30K of data. The original and still standard IBM 8" floppy has a capacity of 256K, so a typist would take about eight and a half working days to fill it with text.

The smallest commercial 5.25" Winchester discs in common use today offer 5 Megabytes (5,000K) of storage. A more usual capacity is 10 Megabytes; this is 40 times the size of the standard IBM floppy, representing about a year's typing. Mini-winnies with the same physical format but with a capacity of 20 Megabytes are fairly common, and current manufacturing techniques have already managed to cram 40 Megabytes into the same sized box.

Forty Megabytes will become a comfortable standard over the next eighteen months. You can think of this as four years' worth of text entry, or something in the region of 85 ordinary length novels.

Perhaps you don't intend spending the next four years wordprocessing away at your micro to type in the whole of **War and Peace** seven times. Maybe your business doesn't actually require you to keep an online database of 150,000 customers (if it does, you've probably outgrown your micro!). In that case you may well be wondering what benefits this superabundance of logical space could possibly bring your way. There are several.



The Micro-Magnum fixed/removable 5¼" Winchester disc system from DMA Systems.

This is a low-cost, fast access 32 Mbyte 8" Winchester sub-system from Newtons Laboratories.

SHARE AND SHARE ALIKE

One popular scheme, particularly in this country, is to allow several users to share the same Winchester disc. This may either be through time sharing on a single processor using operating systems like Digital Research's MP/M, or, increasingly, in configurations that give each user a processor but arrange to have a common data bus connecting them all to the same hard disc.

Systems like this are designed to operate as if each user were working at a stand-alone micro.

The economic advantage of sharing the most expensive component — the Winchester disc itself — is only part of the story. The real gain is that data can easily be shared between users, both at the simple level of leaving messages for each other, as well as allowing all the users to work from and update the same common database.

The logistics of management and manufacturing is often greatly simplified by the multi-user, shared-disc approach. Password protection is usually offered to make sure that there is no unauthorised tampering, and semi-automatic software mechanisms for locking and unlocking records and complete files will also have to be included so that, for example, attempts to update a record while it is being read by another user don't crash the system and destroy the database.

Micros are also beginning to benefit from Winchester drives by looking away from the traditional small-size, small-performance operating systems that have characterised microcomputing since its inception in the early 70s. In place of CP/M and MSDOS, mainframe-inspired operating systems are beginning to appear that do a lot more but demand a lot more space. Unix, the product of Bell Labs, is strongly tipped to come to the fore once Winchester technology becomes commonplace.

Copious fast backing storage is essential to Unix because of its extensive use of overlay-like technique called 'swapping', or 'virtual memory'. A memory manager, built this time into the operating system, constantly juggles chunks of backing store and core memory to create the illusion that core memory is considerably larger than it really is.

Winchesters have their disadvantages. Because the platters are fixed in the drives, a second backing store device, usually a floppy, is essential for getting software in and out of the machine. Archiving requires special consideration; a crashed hard disc that takes 40 Megabytes of your data and programs with it can close down your business if you've omitted to make recent copies. New developments of Winchester-like devices with exchangeable cartridges are beginning to cope with this problem.

WHAT'S IN A NAME

Why call it a Winchester? One much-loved story suggests that the first device IBM produced with this technology had two platters, each with a capacity of 30 Megabytes. The product was dubbed the 3030, and by an extraordinary stroke of fate this was also the name of an early lever-action rifle manufactured by Oliver Fisher Winchester of the Winchester Repeating Arms Company, subsequently famous for the model 73 that helped win the Wild West.

Less romantic chroniclers point out IBM happen to have a plant in a neat New Hampshire town called Winchester. Ah well.

Subscriptions

Personally, we think you'll like our approach to microcomputing. Each month, we invite our readers to join us in an abundance of feature articles, projects, general topics, software listings, news and reviews — all to help committed micro users make more of their microcomputers at home or at work.

However, if you've ever missed a copy of Computing Today on the newstands, you'll not need us to tell you how valuable a subscription can be. Subscribe to CT and for a whole year you can sit back, assured that each issue, lovingly wrapped, will find its way through your letter box.

And it's not difficult! All you have to do is fill in the form below, cut it out and send it (or a photocopy) with your cheque or Postal Order (made payable to ASP Ltd) to:

COMPUTING TODAY Subscriptions,

Infonet Ltd,
Times House,
179 The Marlowes,
Hemel Hempstead,
Herts HP1 1BB.

Alternatively, you can pay by Access or Barclaycard in which case, simply fill in your card number, sign the form and send it off. Please don't send in your card.

Looking for a magazine with a professional approach with material written by micro users for micro users? Why not do yourself a favour and make 1984 the year you subscribe to Computing Today and we'll give you a truly personal approach to microcomputing.

SUBSCRIPTION ORDER FORM

Cut out and SEND TO :
COMPUTING TODAY Subscriptions
INFONET LTD,
TIMES HOUSE,
179 THE MARLOWES,
HEMEL HEMPSTEAD,
HERTS HP1 1BB.

Please commence my subscription to Computing Today with the .... issue.

SUBSCRIPTION RATES (tick ☐ as appropriate)

£13.30 for 12 issues UK ☐
£16.95 for 12 issues Overseas Surface ☐
£36.60 for 12 issues Overseas Air Mail ☐

I am enclosing my (delete as necessary) cheque/ Postal Order/ International Money Order for £..........
(made payable to ASP Ltd)
or
Debit my Access/ Barclaycard*
(*delete as necessary)

BARCLAYCARD VISA

Please use BLOCK CAPITALS and include postcodes.

NAME (Mr/ Mrs/ Miss) ....
delete accordingly

ADDRESS ....

....

.... POSTCODE ....

Signature ....

Date ....

CT June '84

EXTENDING THE 64'S BASIC PART 2

Tony Cross

Last month we talked about the Commodore 64's tokens — now it's time to introduce your own. This month's listing provides the complete skeleton required to flesh out the 64's BASIC.

In last month's article I explained how CBM BASIC converts the keyword names into single byte values called tokens, and that by using tokenised keywords the Interpreter could be made to run much faster. This month I am going to show you how you can add your own keywords to the existing Commodore 64 BASIC.

Adding extra keywords to BASIC is actually fairly easy: it's only necessary to interrupt the Interpreter as it's reading a keyword and check to see if it is one of the new ones. However, as we saw last month, asking the Interpreter to decode the keywords slows it down considerably. The obvious answer, therefore, is to tokenise the new keywords so that they execute as fast as the existing ones. And this is the approach that I am going to be describing this month.

What I have actually done, is to sort out all the additions and modifications that need to be made to the existing BASIC so that you can add new tokenised keywords, and for want of a better name, I called the resulting package the 'Extendable BASIC System'.

EXTENDABLE BASIC

The complete assembly listing of the Extendable BASIC System is given in Listing 1. It's actually a set of separate routines, and the easiest way to see how the whole thing works is to look at each routine separately.

The listing begins with two tables, KEYTAB and ADSTAB, and these are the 'reserved words list' and the 'routine address list' for the new keywords.

KEYTAB (KEYword TABle), located at address $C000, is a 256-byte table for the new keyword names. As with the normal reserved words list, each keyword is entered in ASCII format with the high bit of the last byte set to 1. For example, the keyword name TEST should be entered as $54 $45 $53 $D4. In addition, the last byte in the table must be a null byte, and in order to arrange that this happens 'automatically', KEYTAB is filled with nulls to start with.

ADSTAB (ADdresS TABle), located at address $C100, is a 256-byte table for the new keyword routine addresses. The routine addresses must be in the same order as the keyword names in KEYTAB, and they must be entered in standard low/high format. For example, a routine address of $C432 should be entered as $32 $C4. In addition, all unused addresses must be set to point to address $AF08, which is the address of a routine that prints the SYNTAX ERROR message. To ensure that this happens 'automatically' ADSTAB has been filled with $AF08 bytes to start with.

The next four items are data tables and variables. INTMSG (IniTialisation MeSaGe) is the text of a new start up message, and MODTAB (MODification TABle) and VECTAB (VECtor TABle) are machine code modifications which must be copied into the existing BASIC ROM/RAM. The next item, JMPVEC (JuMP VECtor) is a simple two byte variable.

The next fairly large chunk is the initialisation routines. An initialisation routine is needed because there are a number of modifications which need to be made to the BASIC ROM. (I'll describe what they are and why they are needed later on). The ROM can be modified by copying it into the RAM under neath and then modifying that. In addition, some BASIC routine jump vectors also need changing.

All these changes are carried out by a routine called UNIVST (UNIVersal STart-up routine). In addition to making the ROM-RAM changes it sets the screen background and border colours to black, the pen colour to green and enables the repeat function on all keys. (I prefer the system set up this way and I think you will too.).

UNIVST is called by the two start-up routines WARM and COLD. WARM is the warmstart entry point (SYS 49803 from standard BASIC) and it simply installs the new routines (by calling UNIVST) and then warmstarts BASIC. Any programs in memory are unaffected by a call to WARM.

COLD is the coldstart entry point (SYS 49800 from BASIC). It installs the new routines (by calling UNIVST), clears the screen, prints a new start up message and then coldstarts BASIC. Any programs in memory are 'erased' by a call to COLD.

ROM CHANGES

Now we come to the meat of the system, the modified BASIC ROM routines. GENTOK (GENerate TOKens) is the routine which converts the new keywords into tokens. Obviously, it would have been nice to get the existing tokenise routine to do the conversions for us. Unfortunately, the existing routine cannot deal with a split reserved words list (part in the BASIC ROM and part at address $C000). GENTOK works by comparing each word in the input buffer with the keywords in KEYTAB. If a match is found then the new token is calculated and substituted for the keyword. When the whole line has been scanned, control is returned to the existing tokenise routine which converts any existing keywords into their token values. This gives a rather neat two-stage tokenise process which works very well.

There were one or two problems which needed solving in order to achieve this. First, a small modification had to be made to the existing tokenise routine to prevent it from ignoring the new tokens (MODTAB takes care of this by writing a new branch instruction around this bit of code and filling the rest of it with NOP instructions).

Second and more important, I had to decide what to use for tokens, and in order to keep the new ones completely separate from the existing ones I adopted a two-byte token for the new keywords. Each new keyword token is prefixed by an $FE byte (I would have liked to have used $FF, but that is the current token

Listing 1. Complete assembler listing for the CBM 64 Extendable BASIC System. With this installed you can append any keywords you wish to your BASIC.

```
  10 033C              !###################################
  20 033C              !#                                 #
  30 033C              !# CBM64 - EXTENDABLE BASIC SYSTEM #
  40 033C              !#                                 #
  50 033C              !#     VERSION 3.3 -- 13/01/84     #
  60 033C              !#                                 #
  70 033C              !#  COPYRIGHT (C)  A.L.CROSS 1984  #
  80 033C              !#                                 #
  90 033C              !###################################
 100 033C              !
 110 033C              !
 120 033C              !VARIABLES AND EQUATES
 130 033C              !
 140 033C              KEYCNT     =  $0B
 150 033C              ROMPTR     =  $22
 160 033C              TEMPTR     =  $71
 170 033C              TXTPTR     =  $7A
 180 033C              CHRGET     =  $0073
 190 033C              BUFFER     =  $0200
 200 033C              BASRTN     =  $A7E9
 210 033C              EOLSCN     =  $A909
 220 033C              SYSMSG     =  $E49A
 230 033C              !
 240 033C              !
 250 C000              *=$C000
 260 C000              !
 270 C000              !KEYTAB - 256 BYTE EXTENDED KEYWORD
 280 C000              !NAME TABLE. THE HIGH BIT OF THE
 290 C000              !LAST BYTE IN EACH KEYWORD NAME
 300 C000              !MUST BE SET. THE LAST BYTE IN THE
 310 C000              !TABLE MUST ALWAYS BE $00.
 320 C000              !
 330 C000 000000 KEYTAB      BYT $00,$00,$00,$00
 340 C004 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 350 C00C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 360 C014 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 370 C01C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 380 C024 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 390 C02C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 400 C034 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 410 C03C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 420 C044 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 430 C04C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 440 C054 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 450 C05C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 460 C064 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 470 C06C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 480 C074 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 490 C07C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 500 C084 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 510 C08C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 520 C094 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 530 C09C 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 540 C0A4 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 550 C0AC 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 560 C0B4 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 570 C0BC 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 580 C0C4 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 590 C0CC 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 600 C0D4 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 610 C0DC 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 620 C0E4 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 630 C0EC 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 640 C0F4 000000             BYT $00,$00,$00,$00,$00,$00,$00,$00
 650 C0FC 000000             BYT $00,$00,$00,$00
 660 C100              !
 670 C100              !
 680 C100              !ADSTAB - 256 BYTE EXTENDED KEYWORD
 690 C100              !ADDRESS TABLE. ALL UNUSED BYTES
 700 C100              !MUST POINT TO THE SYNTAX ERROR
 710 C100              !ROUTINE AT ADDRESS $AF08.
 720 C100              !
 730 C100 08AF08 ADSTAB      WOR $AF08,$AF08,$AF08,$AF08
 740 C108 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 750 C110 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 760 C118 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 770 C120 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 780 C128 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 790 C130 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 800 C138 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 810 C140 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 820 C148 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 830 C150 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 840 C158 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 850 C160 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 860 C168 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 870 C170 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 880 C178 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 890 C180 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 900 C188 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 910 C190 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 920 C198 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 930 C1A0 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 940 C1A8 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 950 C1B0 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 960 C1B8 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 970 C1C0 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 980 C1C8 08AF08             WOR $AF08,$AF08,$AF08,$AF08
 990 C1D0 08AF08             WOR $AF08,$AF08,$AF08,$AF08
1000 C1D8 08AF08             WOR $AF08,$AF08,$AF08,$AF08
1010 C1E0 08AF08             WOR $AF08,$AF08,$AF08,$AF08
1020 C1E8 08AF08             WOR $AF08,$AF08,$AF08,$AF08
1030 C1F0 08AF08             WOR $AF08,$AF08,$AF08,$AF08
1040 C1F8 08AF08             WOR $AF08,$AF08,$AF08,$AF08
1050 C200              !
1060 C200              !
1070 C200 930D20 INTMSG      BYT $93,$0D,$20,$20,$20,$20
1080 C206 2A2A2A             BYT '*,'*,'*,$20,'C,'B,'M,'6,'4
1090 C20F 202D20             BYT $20,'-,$20,'E,'X,'T,'E,'N,'D
1100 C218 41424C             BYT 'A,'B,'L,'E,$20,'B,'A,'S,'I,'C
1110 C222 202A2A             BYT $20,'*,'*,'*,$0D,$0D,$00
1120 C229              !
1130 C229 3040EA MODTAB      BYT $30,$40,$EA,$EA,$EA,$EA
1140 C22F EA                 BYT $EA
1150 C230              !
1160 C230 03C382 VECTAB      WOR GENTOK,PRTTOK,EXECUT
1170 C236 C9C3               WOR ARMELT
1180 C238              !
1190 C238 0000   JMPVEC      WOR $0000
1200 C23A              !
1210 C23A              !
1220 C288              *=$C288
1230 C288              !
1240 C288              !COLD - THIS IS THE 'COLD START'
1250 C288              !JUMP VECTOR (SYS 49800).
1260 C288              !
1270 C288 4C94C2 COLD        JMP COLDST
1280 C28B              !
1290 C28B              !
1300 C28B              !WARM - THIS IS THE 'WARM START'
1310 C28B              !ROUTINE (SYS 49803).
1320 C28B              !
1330 C28B 20ABC2 WARM        JSR UNIVST            !INSTALL ROUTINES
1340 C28E 207AA6             JSR $A67A             !RESET STACK
1350 C291 4C74A4             JMP $A474             !WARMSTART BASIC
1360 C294              !
1370 C294              !
1380 C294              !COLDST - THIS IS THE 'COLD START'
1390 C294              !ROUTINE. IT INSTALLS THE NEW
1400 C294              !ROUTINES AND THEN PRINTS THE NEW
1410 C294              !START UP MESSAGE AND EXECUTES THE
1420 C294              !'NEW' COMMAND.
1430 C294              !
1440 C294 20ABC2 COLDST      JSR UNIVST            !INSTALL ROUTINE
1450 C297 A900               LDA #<INTMSG          !PRINT "CBM64...."
1460 C299 A0C2               LDY #>INTMSG
1470 C29B 201EAB             JSR $AB1E
1480 C29E A9A7               LDA #>BASRTN          !PUSH RETURN ADDRESS
1490 C2A0 48                 PHA                   !ONTO THE STACK.
1500 C2A1 A9E9               LDA #<BASRTN
1510 C2A3 48                 PHA
1520 C2A4 A99A               LDA #<SYSMSG          !PRINT "64K...." AND
1530 C2A6 A0E4               LDY #>SYSMSG          !FREE SPACE MESSAGE.
1540 C2A8 4C2DE4             JMP $E42D
1550 C2AB              !
1560 C2AB              !
1570 C2AB              !UNIVST - THIS COPIES THE BASIC
1580 C2AB              !ROM TO THE RAM UNDERNEATH AND
1590 C2AB              !INSTALLS THE NEW ROUTINES. IT IS
1600 C2AB              !CALLED BY WARM AND COLDST
1610 C2AB              !
1620 C2AB A900   UNIVST      LDA #$00              !SET UP ROMPTR
1630 C2AD 8522               STA ROMPTR
1640 C2AF A9A0               LDA #$A0
1650 C2B1 8523               STA ROMPTR+1
1660 C2B3 A000               LDY #$0               !INITIALISE INDEX
1670 C2B5 B122   ROMLP       LDA (ROMPTR),Y        !COPY ROM TO
1680 C2B7 9122               STA (ROMPTR),Y        !RAM UNDERNEATH.
1690 C2B9 C8                 INY
1700 C2BA D0F9               BNE ROMLP             !REPEAT 256 TIMES
1710 C2BC E623               INC ROMPTR+1          !NEXT 256 BYTES
1720 C2BE A523               LDA ROMPTR+1
1730 C2C0 C9C0               CMP #$C0              !END OF BASIC ROM?
1740 C2C2 D0F1               BNE ROMLP
1750 C2C4 A501               LDA $01               !GET I/O REGISTER
1760 C2C6 29FE               AND #$FE              !SWITCH BASIC ROM OUT
1770 C2C8 8501               STA $01
1780 C2CA A207               LDX #$7               !MODIFY THE CBM BASIC
1790 C2CC A000               LDY #$0               !TOKENISE ROUTINE.
1800 C2CE B929C2 MODLP       LDA MODTAB,Y
1810 C2D1 9987A5             STA $A587,Y
1820 C2D4 C8                 INY
1830 C2D5 CA                 DEX
1840 C2D6 D0F6               BNE MODLP
1850 C2D8 A9E5               LDA #<IFTHEN          !MODIFY THE IF..THEN
1860 C2DA 8D49A9             STA $A949             !ROUTINE.
1870 C2DD A9C3               LDA #>IFTHEN
1880 C2DF 8D4AA9             STA $A94A
1890 C2E2 A208               LDX #$8               !MODIFY THE BASIC
1900 C2E4 A000               LDY #$0               !JUMP VECTORS.
1910 C2E6 B930C2 VECLP       LDA VECTAB,Y
1920 C2E9 990403             STA $0304,Y
1930 C2EC C8                 INY
1940 C2ED CA                 DEX
1950 C2EE D0F6               BNE VECLP
1960 C2F0 A905               LDA #$5               !SET PEN TO GREEN
1970 C2F2 8D8602             STA $0286
1980 C2F5 A900               LDA #$0               !SET SCREEN & BORDER
1990 C2F7 8D20D0             STA $D020             !TO BLACK
2000 C2FA 8D21D0             STA $D021
2010 C2FD A980               LDA #$80              !REPEAT ALL KEYS
2020 C2FF 8D8A02             STA $028A
2030 C302 60                 RTS
2040 C303              !
2050 C303              !
2060 C303              !GENTOK - THIS IS THE NEW PART OF
2070 C303              !THE CRUNCH TOKEN ROUTINE. IT
2080 C303              !SCANS THE INPUT BUFFER AND
2090 C303              !CONVERTS ALL NEW KEYWORDS TO
2100 C303              !THEIR TOKENS (WHICH ARE PREFIXED
2110 C303              !BY AN $FE CHARACTER). IT THEN
2120 C303              !'FALLS THROUGH' TO THE MODIFIED
2130 C303              !CBM BASIC CRUNCH TOKENS ROUTINE.
2140 C303              !
2150 C303 A57A   GENTOK      LDA TXTPTR            !SAVE CURRENT
2160 C305 48                 PHA                   !TEXT POINTER VALUE
2170 C306 A57B               LDA TXTPTR+1
2180 C308 8572               STA TEMPTR+1
2190 C30A A000   MAINLP      LDY #$0               !INIT TABLE INDEX
2200 C30C A67A               LDX TXTPTR            !INIT BUFFER INDEX
2210 C30E A901               LDA #$01
2220 C310 850B               STA KEYCNT            !INIT KEYWORD COUNT
2230 C312 88     INRLP       DEY
2240 C313 CA                 DEX
2250 C314 E8     SCANLP      INX                   !NEXT CHAR IN BUFFER
2260 C315 C8                 INY                   !NEXT CHAR IN TABLE
2270 C316 BD0002             LDA BUFFER,X          !GET NEXT BUFFER CHAR
2280 C319 C922               CMP #$22              !IS IT A QUOTE?
2290 C31B D010               BNE NOTQTE
2300 C31D E8     QUOTLP      INX                   !NOW SCAN FOR ANOTHER
2310 C31E BD0002             LDA BUFFER,X          !QUOTE CHARACTER
2320 C321 F059               BEQ TOKEND            !(OR AN END OF LINE)
2330 C323 C922               CMP #$22
2340 C325 D0F6               BNE QUOTLP
2350 C327 E8                 INX                   !STEP OVER QUOTE
2360 C328 867A               STX TXTPTR
2370 C32A 38                 SEC
2380 C32B B0DD               BCS MAINLP
2390 C32D 38     NOTQTE      SEC
2400 C32E F900C0             SBC KEYTAB,Y          !AND NEXT TABLE CHAR
2410 C331 F0E1               BEQ SCANLP            !IF THEY ARE THE SAME
2420 C333 C980               CMP #$80              !& WITH HIGH BIT SET?
2430 C335 D02C               BNE NXTKEY
2440 C337 E8                 INX
2450 C338 8671               STX TEMPTR            !SAVE BUFFER INDEX
2460 C33A A67A               LDX TXTPTR            !BACK TO START
2470 C33C A9FE               LDA #$FE
2480 C33E 9D0002             STA BUFFER,X          !POKE FIRST TOKEN IN
2490 C341 E8                 INX
2500 C342 A50B               LDA KEYCNT
2510 C344 9D0002             STA BUFFER,X          !POKE 2ND BYTE IN
2520 C347 E8                 INX                   !UPDATE TEXT POINTER
2530 C348 8A                 TXA
2540 C349 48                 PHA                   !AND SAVE IT
2550 C34A A571               LDA TEMPTR
2560 C34C 857A               STA TXTPTR
2570 C34E 2009A9             JSR EOLSCN            !LENGTH OF LINE LEFT
2580 C351 C8                 INY
2590 C352 68                 PLA
2600 C353 857A               STA TXTPTR            !UPDATE TEXT POINTER
2610 C355 98                 TYA
2620 C356 AA                 TAX
2630 C357 A000               LDY #$0               !INITIALISE INDEX
2640 C359 B171   MOVELP      LDA (TEMPTR),Y        !CLOSE UP
2650 C35B 917A               STA (TXTPTR),Y        !THE UNWANTED GAP.
2660 C35D C8                 INY
2670 C35E CA                 DEX
2680 C35F D0F8               BNE MOVELP
2690 C361 F0A7               BEQ MAINLP            !SCAN FOR NEXT KEYWORD
2700 C363 A67A   NXTKEY      LDX TXTPTR            !BACK TO START
2710 C365 E60B               INC KEYCNT            !BUMP KEYWORD COUNTER
2720 C367 C8     KEYSCN      INY
2730 C368 B9FFBF             LDA KEYTAB-1,Y        !MOVE TABLE INDEX
2740 C36B 10FA               BPL KEYSCN            !TO END OF KEYWORD
2750 C36D B900C0             LDA KEYTAB,Y          !IF NOT AT END OF
2760 C370 D0A0               BNE INRLP             !TABLE THEN JUMP BACK
```

```
C372 BD0002            LDA BUFFER,X       !IF NOT AT END OF
C375 F005              BEQ TOKEND         !LINE THEN REPEAT
C377 E67A              INC TXTPTR         !ADVANCE TEXT POINTER
C379 38                SEC
C37A B08E              BCS MAINLP         !AND REPEAT
C37C 68     TOKEND     PLA                !RESTORE TEXT POINTER
C37D 857A              STA TXTPTR
C37F 4C7CA5            JMP $A57C          !RETURN TO BASIC
C382        !
C382        !
C382        !PRTTOK - THIS IS THE NEW PART OF
C382        !THE EXPAND AND PRINT KEYWORD
C382        !TOKENS ROUTINE. IF THE CURRENT
C382        !TOKEN IS FOR A NEW KEYWORD
C382        !(PREFIXED BY $FE) THEN THIS
C382        !ROUTINE EXPANDS AND PRINTS IT.
C382        !
C382 C9FE   PRTTOK     CMP #$FE           !NEW KEYWORD?
C384 F003              BEQ LSTKEY
C386 4C1AA7            JMP $A71A          !IF NOT GO BACK
C389 C8     LSTKEY     INY
C38A B15F              LDA ($5F),Y        !GET NEXT TOKEN BYTE
C38C 8449              STY $49
C38E A0FF              LDY #$FF           !INIT TABLE POINTER
C390 AA                TAX
C391 CA                DEX                !INIT KEYWORD COUNTER
C392 F009              BEQ KEYPRT         !IF FIRST KEYWORD
C394 C8     KEYLP      INY
C395 B900C0            LDA KEYTAB,Y       !STEP OVER KEYWORD
C398 10FA              BPL KEYLP
C39A CA                DEX                !X NUMBER OF TIMES
C39B D0F7              BNE KEYLP
C39D C8     KEYPRT     INY
C39E B900C0            LDA KEYTAB,Y
C3A1 3005              BMI BASPRT         !IF HIGH BIT SET
C3A3 2047AB            JSR $AB47          !PRINT CHARACTER
C3A6 D0F5              BNE KEYPRT
C3A8 4CEFA6 BASPRT     JMP $A6EF          !PRINT LAST CHAR
C3AB        !
C3AB        !
C3AB        !EXECUT - THIS IS THE NEW PART OF
C3AB        !THE STATEMENT EXECUTION ROUTINE.
C3AB        !IF THE CURRENT TOKEN IS FOR A NEW
C3AB        !KEYWORD (PREFIXED BY $FE) THEN
C3AB        !THIS ROUTINE EXECUTES THE
C3AB        !APPROPRIATE STATEMENT KEYWORD
C3AB        !ROUTINE.
C3AB        !
C3AB 207300 EXECUT     JSR CHRGET         !GET NEXT CHAR
C3AE 08                PHP                !SAVE FLAGS
C3AF C9FE              CMP #$FE           !NEW KEYWORD?
C3B1 F004              BEQ EXEC
C3B3 28                PLP                !RESTORE FLAGS
C3B4 4CE7A7            JMP $A7E7          !GO BACK IF NOT
C3B7 28     EXEC       PLP                !SCRAP OLD FLAGS
C3B8 20FAC3            JSR TOKGET         !GET NEXT TOKEN
C3BB A9A7              LDA #>BASRTN       !PUSH RETURN
C3BD 48                PHA                !ADDRESS ON THE STACK
C3BE A9E9              LDA #<BASRTN
C3C0 48                PHA
C3C1 207300            JSR CHRGET         !GET FIRST PARAMETER
C3C4 A200              LDX #$00           !SET STATEMENT FLAG
C3C6 6C38C2            JMP (JMPVEC)       !EXECUTE STATEMENT
C3C9        !
C3C9        !
C3C9        !ARMELT - THIS IS THE NEW PART OF
C3C9        !THE GET AN ARITHMETIC ELEMENT
C3C9        !INTO FPA1 ROUTINE. IF THE CURRENT
C3C9        !TOKEN IS FOR A NEW KEYWORD
C3C9        !(PREFIXED BY $FE) THEN THIS
C3C9        !ROUTINE EXECUTES THE APPROPRIATE
C3C9        !FUNCTION KEYWORD ROUTINE.
C3C9        !
C3C9 A900   ARMELT     LDA #$0            !SET STRING FLAG
C3CB 850D              STA $0D            !TO NOT STRING.
C3CD 207300            JSR CHRGET         !GET NEXT CHARACTER
C3D0 08                PHP                !SAVE FLAGS
C3D1 C9FE              CMP #$FE           !NEW KEYWORD?
C3D3 F004              BEQ ARITHM
C3D5 28                PLP                !RESTORE FLAGS
C3D6 4C8DAE            JMP $AE8D          !RETURN TO BASIC
C3D9 28     ARITHM     PLP                !SCRAP OLD FLAGS
C3DA 20FAC3            JSR TOKGET         !GET NEXT TOKEN
C3DD 207300            JSR CHRGET         !GET FIRST PARAMETER
C3E0 A2FF              LDX #$FF           !SET FUNCTION FLAG
C3E2 6C38C2            JMP (JMPVEC)       !EXECUTE FUNCTION
C3E5        !
C3E5        !
C3E5        !IFTHEN - THIS IS THE MODIFIED
C3E5        !END OF THE IF..THEN ROUTINE. IF
C3E5        !THE CURRENT CONTENTS OF THE A
C3E5        !REGISTER IS A NEW KEYWORD TOKEN
C3E5        !(PREFIXED BY $FE) THEN THIS
C3E5        !ROUTINE EXECUTES THE APPROPRIATE
C3E5        !STATEMENT KEYWORD ROUTINE.
C3E5        !
C3E5 08     IFTHEN     PHP                !SAVE FLAGS
C3E6 C9FE              CMP #$FE           !NEW KEYWORD?
C3E8 F004              BEQ DOIF
C3EA 28                PLP                !RESTORE FLAGS
C3EB 4CEDA7            JMP $A7ED          !RETURN TO BASIC
C3EE 28     DOIF       PLP                !SCRAP OLD FLAGS
C3EF 20FAC3            JSR TOKGET         !GET NEXT TOKEN
C3F2 207300            JSR CHRGET         !GET FIRST CHARACTER
C3F5 A200              LDX #$00           !SET STATEMENT FLAG
C3F7 6C38C2            JMP (JMPVEC)       !EXECUTE ROUTINE
C3FA        !
C3FA        !
C3FA        !TOKGET - THIS IS A SUBROUTINE
C3FA        !THAT GETS THE SECOND TOKEN BYTE
C3FA        !AND USES IT AS A POINTER INTO
C3FA        !ADSTAB. THE VALUE FOUND HERE IS
C3FA        !THEN LOADED INTO JMPVEC.
C3FA        !
C3FA E67A   TOKGET     INC TXTPTR         !ADVANCE TEXT
C3FC D002              BNE TOKGOT         !POINTER
C3FE E67B              INC TXTPTR+1
C400 A000   TOKGOT     LDY #$00           !INITIALISE INDEX
C402 B17A              LDA (TXTPTR),Y     !GET TOKEN BYTE
C404 38                SEC                !DECREMENT TOKEN
C405 E901              SBC #$01           !VALUE AND THEN
C407 0A                ASL A              !MULTIPLY BY 2
C408 A8                TAY
C409 B900C1            LDA ADSTAB,Y       !GET LOW ADDRESS
C40C 8D38C2            STA JMPVEC
C40F B901C1            LDA ADSTAB+1,Y     !GET HIGH ADDRESS
C412 8D39C2            STA JMPVEC+1
C415 60                RTS
C416        !
C416        !
C416        !THE KEYWORD ROUTINES - START HERE
C416        !
```

for PI). $FE is not used as a token by the existing BASIC, but using it here makes testing for new keyword tokens very simple indeed. The second token byte is calculated, as with existing keywords, from the position of the keyword in KEYTAB, except that the high bit is not set. (Confusion can occur between existing tokens and new tokens if the high bit of the second token byte is set).

For example, the first keyword in the table will be tokenised to $FE $01, and the second to $FE $02 and so on. I had to start the tokens at $01 because $FE $00 is $FE followed by a new line! The only restriction this imposes is that all new keyword names must be at least two characters long (to make room for the two token bytes to be copied in).

This process does create one minor problem, however, and that occurs on lines with REM and DATA statements in them. If a new keyword is included in a REM or a DATA statement then it will be tokenised by GENTOK (at this stage GENTOK doesn't know that this is a REM or DATA statement). This can easily be avoided by ensuring that any new keyword names in a REM or DATA statement are enclosed in quotes (") because GENTOK does not tokenise the contents of quoted strings.

In order to link GENTOK to the existing BASIC routines it is necessary to change all calls to the existing tokenise routine so that they call GENTOK instead. Fortunately, this couldn't be easier because the existing tokenise routine is indirectly addressed at location $0304/$0305. In other words, location $0304/$0305 contains the current address of the tokenise routine, and all calls to the routine are made via this location. All that is needed then, is to make location $0304/$0305 point to the GENTOK routine, and this is done during initialisation.

PRTTOK (PRinT TOKens) is the routine which expands and prints the new keywords during a LIST. It works by first checking that the current token is $FE (new keyword). If not, then control is passed back to the normal LIST routine, otherwise the second token byte is used as a pointer into KEYTAB. The token is actually decremented by one first, because the tokens start at $01 but KEYTAB counts from 0. PRTTOK then prints the keyboard characters from the table until one with the high bit set is found, when it returns to the normal LIST routine which prints this last character. PRTTOK is linked to BASIC in a similar manner to GENTOK, that is, by changing the indirect address at location $0306/$0307 to point to the PRTTOK routine. Once again, this change is done during initialisation.

EXECUTIVE ACTION

EXECUT (EXECUTion) is the routine which interprets and executes the new keywords. It works by first checking that the current token is $FE (new keyword). If not, then control is passed back to the normal execution routine, otherwise the second token byte is used as a pointer into ADSTAB. Again, the token is actually decremented by one first, because the tokens start at $01 but ADSTAB counts from 0. The two-byte address found at this location is copied into the JMPVEC variable and an indirect jump to this location is used to execute the routine.

So that the keyword routines can be ended with a nice neat RTS instruction, the normal execution routine's return address is pushed onto the stack before the jump is made. EXECUT is linked to BASIC just like GENTOK and PRTTOK by changing the indirect jump address at location $0308/$0309 to point to the EXECUT routine. This change is also made during the initialisation process.

The next two routines modify parts of the existing BASIC ROM that I haven't mentioned yet, but their functions should be fairly clear after we have looked at the modifications.

ARMELT (ARithMetic ELemenT) is a routine which is used during the evaluation of expressions — this is something which I'll be looking at in a later article. Briefly, ARMELT is a version of EXECUT which executes functions instead of statements. Functions return a value; for example SIN(30) is a function. Statements carry out some action, so that PRINT "FRED" is a statement. So in the program

10 PRINT SIN(30)

the PRINT keyword will be executed by a routine like EXECUT while the SIN keyword will be executed by a routine like ARMELT.

ARMELT works very much like EXECUT, and is linked to BASIC by changing the indirect jump address at location $030A/$030B. As before, this change is made during initialistaion.

IFTHEN (IF . . . THEN) is a modification to the THEN part of the existing IF . . . THEN construct. The modification is required because the existing THEN clause jumps directly to the existing execution routine, to execute the statement following the THEN. IFTHEN performs this function for the new keywords, and works in a similar manner to EXECUT. Linking IFTHEN to BASIC is slightly different than for the other routines because the existing THEN clause is not indirectly addressed. The change, which is made during initialisation, is to a jump address in the BASIC ROM/RAM.

TOKGET is a subroutine which is used by EXECUT, ARMELT and IFTHEN to get the value of the second token byte. It also decrements the token and uses it as a pointer into ADSTAB. The address found at this location is then loaded into the JMPVEC variable and control is returned to the calling routine.

And finally, we come to the keyword routines themselves, and here it's largely up to you — although later articles will contain some useful new keyword routines which you can use. There are one or two points which you should bear in mind when writing your new keyword routines, and these are as follows:

- All keyword routines must end in a RTS instruction. There must be no other 'way out' of any routines, except via fatal error messages (such as SYNTAX ERROR).
- On entry to the routines the A register will contain the next non-space character after the keyword, and the 'text pointer' (see next month's article) will point to this character.
- On entry to the routines the X register will contain $00 if the routine has been called from EXECUT or IFTHEN (ie it was used as a statement), and $FF if it has been called from ARMELT (ie it was used as a function). This can be used as a test to error illegal function calls.
- The Y register contents on entry to the routines are undefined.
- All the 6510 registers are available for use within the routines, and they need not have any special values on return.
- The 6510 stack and stack pointer can be used freely within the routines provided that, on exit from the routine, the contents of the stack and stack pointer are the same as they were on entry.

USING THE SYSTEM

Don't worry if there are things about the system which you don't understand yet : it really isn't necessary to know anything about how the system works. All you ned to remember in order to be able to add new keywords is:

- Write the keyword routine. The rest of this series will help you do this.
- Locate it somewhere in RAM. The space between $C416 and $CFFF is the best place. Otherwise dropping the top of BASIC will allow you to create as much space as you want.
- Write the keyword name into the next available locations in KEYTAB. Remember to set the high bit of the last byte.
- Write the address of the routine into the next available locations in ADSTAB (low byte first).
- If the Extendable BASIC System is not already installed then either warmstart (SYS 49803) or coldstart (SYS 49800) the system to install the new routines.

The new keyword is now part of the Commodore 64's BASIC, and can be used freely in any valid part of your programs. If you want to save your new version of BASIC, then make sure that you save the whole Extendable BASIC System as well as your new routines. In this way you can simply load and install the new version of BASIC whenever you need it.

NEXT MONTH

You can now add new tokenised keywords to the existing BASIC, where they will execute just as fast as the existing ones. Next month I will show you how to go about writing the new keyword routines, including things like evaluating expressions and checking for special characters. To help you do this I will also be describing some of the many useful routines that you can use in the existing BASIC ROM.

BACKNUMBERS

MARCH 1983
Colour Genie reviewed, Epson HX-20 review, PEEKing the Spectrum, Into Atari's BASIC, Terminology translated.

APRIL 1983
Froglet on the BBC Micro, PC-1251 hand-held review, Valley Variations, Galaxy reviewed, Micro Database, Lower case UK101.

MAY 1983
Spectrum Book Survey, Oric-1 Review, Going FORTH Again, Jupiter Ace review.

JUNE 1983
Interrupt handling, Rubic simulation on the Spectrum, Beating the RS232 Blues, Lynx review, Indexer.

JULY 1983
Atari renumber, 16-bit micros, Bomb-proof Tandy, Olivetti Praxis 30 review, Ikon Hobbit tape drive review.

AUGUST 1983
Speeding up the Sharp, Premier Dragon disc drive, Sord M5 review, BBC String Store, Planetfall.

SEPTEMBER 1983
FELIX knowledge shops, Software protection, Torch disc pack, ZX81 Backgammon, Dragon character generator, Three Tandy computers.

OCTOBER 1983
Slingshot game, Sharp MZ-700 review, Sharp MZ-3541 review, Z80 Disassembler, A better TRSDOS, Improved VIC-20 editor.

NOVEMBER 1983
BBC Word Processor, ZX LPRINT review, Laser 200 review, Writing Adventures, Learning FORTH Part 1, PET tape append.

DECEMBER 1983
MIKRO assembler review, Getting More from the 64 Part 1, Adventures part 2, Curve-fitting, BBC Touch Typing Tutor.

JANUARY 1984
TRS-80 programmer's aid, Apple music, Electron review, TRS-80 screen editor, calendar program.

FEBRUARY 1984
Using MX-80 graphics, Colour Genie monitor, non-random random numbers, ZX81-FORTH, Program recovery on the Commodore 64.

MARCH 1984
Easycode part 1, BBC poker, Spectrum SCOPE review, Genie utilities, Spectrum Centronics interface.

APRIL 1984
Memotech MTX500 review, Genie BASIC extensions, Brainstorm review, Disassembly techniques, Recursion.

If you've lost, lent or had stolen one of those precious back copies of Computing Today then now is your chance to fill the gap in your collection. The list of issues given here represents the few remaining copies that we have available to help complete your library of all that's good in features, programs and reviews.

If you want one of these issues, it's going to cost you £1.40 (including postage and packing) but we think that's a small price to pay for the satisfaction you'll get. Ordering could hardly be made simpler — just fill in the form, cut it out (or send a photocopy) together with your money to:

**Backnumbers,
Infonet Ltd,
Times House,
179 The Marlowes,
Hemel Hempstead,
Herts HP1 1BB.**

If you wait until next month to do it, the chances are that we'll have run out of the very issue you wanted!

A Computing today Reader Service

BACKNUMBERS

Please send me the following Backnumbers

ISSUE

At £1.40 each. I enclose £ ....

NAME ....

ADDRESS ....

....

....

.... POSTCODE ....

Signature ....

I enclose a cheque/PO for £.....(Payable to ASP Ltd)
I wish to pay by credit card

Access ☐ Barclaycard ☐

Insert Card No.

If you wish to pay by Access or Barclaycard, just fill in your card number and sign the form, do not send your card.

Please allow 21 days for delivery.

CT June '84

MICROWRITE ON THE BBC

D. S. Peckett

If you can't get to grips with touch-typing, either the two or 10-fingered variety, why not split the difference and use five? We look at the novel one-handed Quinkey keyboard from Microwriter.

Are you a 'peek and poke' typist? Do your thoughts run ahead of you at the keyboard? Do you wish that you could watch the screen rather than the keys as you type your latest masterpiece in to your Beeb? If so, you might like to consider Microwriter's 'Quinkey' keyboard.

In this article I will take a short look at this unusual accessory and give you an impression of my experiences in coming to terms with it. I will go on to attempt to outline its strengths and weaknesses, and try to give an idea of where it might be of most use. First of all, though, what is a Microwriter?

In the late 70s, an ex-patriate American, Cy Endfield, invented a cunning device which was nothing more or less than a single-handed word processor. The device's most remarkable feature was and is that it is operated entirely via six keys. This original Microwriter has enjoyed steady, though subdued, success and many companies and Government departments have issued them to their senior executives. Their big attractions have been that they are easy to learn to use, they are normally quicker than handwriting and that they produce directly a tape which can be loaded into a conventional word processor for printing.

However, a major drawback to these apparently clever and certainly complex devices has been cost. Depending on the precise configuration, a Microwriter will set you back around £400. This puts them out of the reckoning for all but the most well-heeled private individuals.

Although I have no connection whatsoever with the company, I have always been intrigued by the Microwriter concept and, when the Quinkey microwriting keyboard for the Beeb appeared, I decided to investigate further.

WHAT DO YOU GET?

The complete Quinkey package, which costs just over £40, contains the Quinkey itself, an instruction manual, connecting leads, a cassette containing a demo program and two different driver routines and, last but not least, a set of handy reference cards. The Quinkey is about the size of a thick paperback and, as you see, has just six keys on it. It is designed for right-handed operation and, as far as I am aware, does not have a left-handed version. Only sinistrals will know how important that lack may or may not be.

The Quinkey plugs into the computer's analogue input port — so you need a Model B — and requires OS 1.2 to work successfully. The system can handle up to four Quinkeys, which might be very useful in a classroom, and the two driver packages are intended for dif-

ferent situations.

The device works with BASIC I and BASIC II, the Acorn DFS, wordprocessors such as Wordwise and View, and many other important software packages.

Both driver routines are relocatable and take up ¾K of RAM at the bottom of memory, shifting PAGE to suit. "PROG" will handle four keyboards and is intended for general use, while "WP" will only work with one Quinkey and is a faster routine aimed specifically at word processing. The latter routine also provides the special control codes which WPs need — I wrote this review using the Microwriter and Wordwise.

These drivers have one irritating feature — they are not preserved through Break key operations, the computer going back to its normal configuration. I found this most annoying, both in normal operation and when using Wordwise, where the key normally gives a quick way of clearing a document.

I received a provisional copy of the manual, which is a substantial, spiral-bound document of 32 A4-size pages. It contains a useful introduction to the Quinkey, which leads rapidly into its main section, a well-written tutorial on the system's use. The instructions are illustrated by many little cartoons, which will appeal or appal, depending on your taste — I liked them.

Once you have mastered the Quinkey, the manual remains useful as a reference document and as a set of instructions on how to adapt the driver routines to your specific needs. I found this technical section comprehensive, although not particularly easy to follow.

FIVE FINGER EXERCISES

Enough of this — how on earth do you work the beast? The top five keys are the main inputs, while the bottom left one is used as a combined shift and control key to select various operating modes. The real secret of the keyboard's success and simplicity lies, however, in the superb teaching approach, which combines pictures, clever mnemonics and some appalling visual puns to drive the keying patterns into your head very quickly.

As far as possible, the keys used for each letter have been carefully selected to mimic the letter's shape. These are reinforced by phrases and diagrams, such as those accompanying this article, which make learning quick and easy. Who can forget such tricks as 'complete Press for "P" or 'Top of the "T" '?

When operating normally, the keyboard produces lowercase letters and the main punctuation marks, while a single press on the lower-left 'command key' makes the next character a capital. A double press puts the system into UPPER-CASE only. Pressing the two lower left keys together returns the Quinkey to its normal lower-case mode at any time.

To obtain numbers and the less common punctuation, press the command key together with the keys normally covered by the middle and ring fingers. As before, one press gives one character while a double press locks into the new range. Once in the number set a whole new range of mnemonics come into play but they are very cleverly linked to the standard letter keys. For instance, a question mark comes from the same

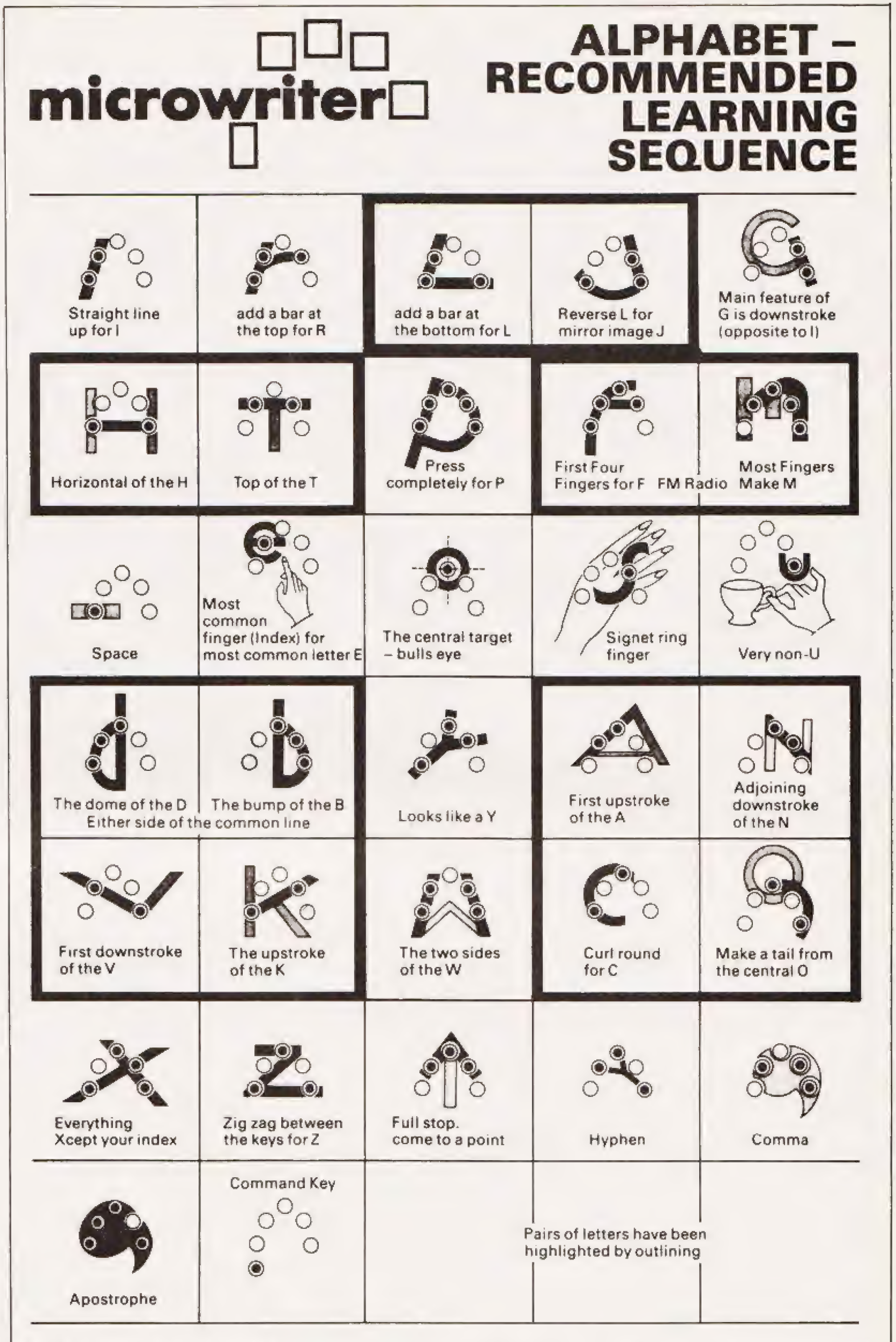


Fig. 1 Microwriter mnemonics

Schools would probably find the Quinkey a great help — several children can interact on one BBC Micro, and youngsters would pick up the keying sequences very quickly.

combination that normally gives "Y" (for "why?"!). Corny but brilliant.

Other keying sequences give you the special combinations needed for word processing and editing, to emulate the red function keys, etc. I do feel, however, that some of the sequences seem contrived and rather involved, and I still find it easier to use the ordinary keyboard for them. The unadventurous might be pleased to hear that the Quinkey does not disable the Beeb's normal keyboard. In particular, it is useful to be able to press cursor and function keys with the left hand while operating the Quinkey with the right.

The upshot of this combination of good design and clever tutorial is that it is remarkably easy to learn to microwrite. I learnt the alphabet, numbers and the main punctuation symbols in an hour, and I understand that that is nothing special. The record stands at something less than 10 minutes!

Once you know how to drive the machine, the rest comes down to practice. I can now use the Microwriter rather faster than writing by hand, although I have yet to reach the 45 wpm which most experienced users get to. For comparison, that is about the speed of a pretty good touch typist.

In use, the system has proved to be totally reliable, although sometimes a little confused by my house's exceptionally dirty mains supply. That's really a personal problem, though.

TARGET AUDIENCE

I am not totally sure at whom the Quinkey is aimed. It is a little pricy for the 'typical' home computer owner (if such a beast exists) and it does take a little perseverance to learn to use it. It takes a great deal more effort to learn to touch type, however. It would be very valuable to anyone who does more than a modest amount of word processing on the Beeb, and also to prolific programmers. When programming, however, it is sometimes a little irritating to have to keep changing from letter to number shift and vice-versa; fast two-finger typists probably would not see much of a benefit from the Microwriter in this area.

The Quinkey could also be very useful in schools, where the ability to connect four keyboards to a single computer might be most valuable. I rather think that children would take to microwriting like ducks to water.

In any application, a major benefit of the device is that it makes it simple to watch the screen rather than the keyboard. This makes it much easier to spot and correct errors as they happen. In fact, I find it virtually impossible to watch the Quinkey when I am using it — I get as confused as the centipede who tried to work out how he moved his legs.

The Microwriter could make some very novel games possible, since it is rather easier to use it to provide multiple functions than to do the same thing via the keyboard. A joystick would often be even better, however.

CONCLUSION

The Microwriter is a very nicely made and cunning accessory for the BBC Computer. I believe that it would be of most value as a supplement, rather than a replacement, for the normal keyboard. Its clever documentation makes it very easy to learn to drive and, if you do any more than the smallest amount of word processing, well worth persevering with.

It would also be useful to programmers who cannot come to terms with the traditional QWERTY keyboard, although the complexity of some of the mode changes do slow it down a little in this role.

Schools might like to consider using it, particularly since it makes it simple to let four children (five if you include the normal keyboard) interact on a single computer. The company offer several games and other programs for multi-user systems, and these might be very interesting to schools.

Is it worth buying? Probably "yes", but it depends on your application — it won't solve all your data entry problems.

A version of the Quinkey is planned for the Spectrum; that could be interesting and will certainly increase the device's market significance. For more information, contact Microwriter at 31 Southampton Row, London WC1 (phone 01-831 6801).

THE LEECH

Richard Sargent

Program development on the Spectrum is no fun with that rubber keyboard. Why not develop your machine code on another machine, then download into the Spectrum like the commercial software houses do? The Leech will make this possible — let the blood-taking begin . . .

The Leech is a program which enables one computer to communicate with another and it is primarily intended for use with the ZX Spectrum (16K), although it can, of course, be used with a 48K Spectrum. Leeches are fairly lowly creatures and some critics would say that the 16K Spectrum is way down in the computer hierarchy too — after all, you soon run out of memory fairly quickly and if you put a machine code monitor or (worse still) an editor/assembler into your machine, there is hardly enough space left to enter "Copyright T.Bloggs 1984", let alone the actual program!

The solution is to put the Leech into the 256 bytes of the Spectrum printer buffer and pull your code (data files or machine-code) across from another computer. In this way you can do your editing and assembling on a friend's machine or on a machine at the office or at school, which hopefully has rather more memory than your Spectrum. The machine must also possess a Centronics or other similar parallel output port.

THE CENTRONICS PORT

The standard Centronics port provides at least nine lines of output and at least one line of input which can be reliably found on certain of the 36 pins of the Centronics plug. Table 1 shows the pins from the Spectrum's viewpoint: the Centronics nomenclature is given in brackets.

CODE TRANSFER

The object of the exercise is to be able to pass the code into the Spectrum at a reasonably fast rate (ie virtually instantaneously) and with 100% reliability. The code should be able to come from any part of the host computer's memory and be loaded into any part of the Spectrum's memory, and a warning should be given if poor reception is suspected. The Leech code is sensibly tucked away in the Spectrum printer buffer and since this means that the ZX Printer cannot be used without scrambling the Leech code itself, a small printer driver has been written so that (with the addition of the appropriate leads), the Leech hardware may be used to drive an alternative printer.

THE HARDWARE

The add-on board is extremely simple and can be built for about £5, but the total cost will vary according to the socket on the host computer. A Centronics type socket alone can cost £5.50, but you may be able to manage without one. The cost of plugs and cables always seems to be on the high side, but you may console yourself in the knowledge that the cost of the entire project with gold-plated sockets and a PCB will amount to less than the asking price for the RS232 cable which plugs into Sinclair's Interface One!

The Z80A PIO requires no special handling precautions, but be careful to bend the pins into the vertical plane so that the IC crunches firmly into its 40 pin socket. Be sure to obtain the Z80A PIO — "A" for 4 MHz — since the slower variety probably will not work on the Spectrum's 3.5 MHz clock. If you are building the unit on Veroboard it is always worth double-checking your wiring, playing attention in particular to the decode lines A8 and A9. Reversal of these can cause the software to initialise the PIO incorrectly, setting lines that should be inputs into outputs and doing costly damage not only to the PIO itself but also to the output components of the host computer.

RUNNING THE LEECH

Both the receiving and transmitting programs are written in Z80 machine code, for a number of good reasons. The Spectrum code can't be in BASIC because it is destined for the printer-buffer memory space, but that apart, machine code offers the facility to manipulate the parallel ports directly, to disable software-generated interrupts (which would complicate timing calculations) and to run at top speed. A short BASIC program has been written for the host computer but this only puts a check-list of the setting up procedure on the screen and is independent of the transfer program.

The sequence of operation is critical to the safety of the PIO chips and to the reliability of the transfer of bytes — as explained above the versatile PIOs can have their lines con-

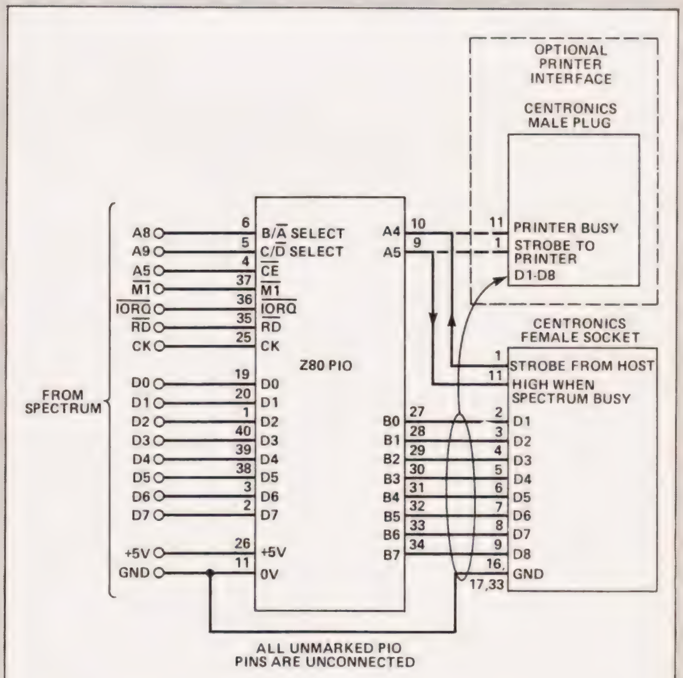


Fig. 1 Circuit diagram of the required hardware.

TABLE 1

Pin 1	Control input (strobe)
Pins 2-9	Eight data inputs (data bits)
Pin 10	Control output (acknowledgé)
Pin 11	Control output (busy)
Pin 16	0 Volts common

If pin 10 is connected, pin 11 will be unconnected (and vice versa).

figured in a variety of ways and so they must be initialised properly *before* physically joining the two computers. The procedure is as follows:

- Stage 1 — computers not linked.
- Stage 2 — LOAD Leech programs.
- Stage 3 — RUN host computer program (sets host PIO)
- Stage 4 — RUN Spectrum code (sets Leech PIO).
- Stage 5 — Link computers together.
- Stage 6 — Press key on Spectrum.
- Stage 7 — Press key on host computer.
- Stage 8 — Verify checksums agree. JOB DONE.

TIMING CONSIDERATIONS

The two programs keep in step with one another using the two control lines of the Centronics interface — Strobe and Busy (or Acknowledge). The flow diagrams in Fig. 2 show the core of the transmit/receive code. The delay block in the Spectrum code allows the host computer to latch on to the Spectrum 'not busy' signal while the delay block in the host computer's code allows the Spectrum to latch on to the host computer's 'strobe'. If you are transmitting between computers with widely different CPU clock frequencies it would be wise to lengthen the delay within the *faster* computer. The values given worked perfectly with a host computer with a CPU clock frequency of 4MHz.

THE BASIC LISTING

The actual form of this program will depend upon the commands of the host computer's BASIC and the nature of its Centronics port. The chances are that the port will consist of cheap TTL chips and since these do not need initialising line 20 will not be needed. The POKEs of lines 36-42 pass values to the machine code. The prompts should be followed in strict order and line 80 actually runs the machine code (Leech Listing 3). The form of line 80 will vary but you should try to get the checksum printed on the screen. Some BASICS allow a return from machine code to include a numerical value (normally expected in the A,B or B,C register pair). If this is not so, the checksum value must be picked up by a PEEK and this is done in line 90. Needless to say the screened value should tally with that screened by the Spectrum. If it does not the Spectrum bytes have become corrupted and another transfer attempt must be made.

Just to make life difficult, the number returned by the host

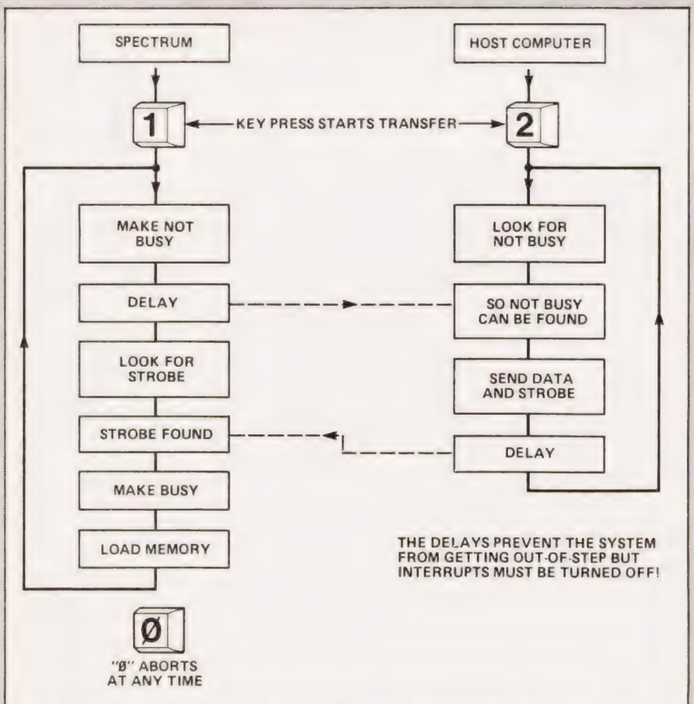


Fig. 2 Flowchart for the two programs (Listings 1 and 2).

```
10 PRINT "Computers should be unlinked"
20 REM Set host PIO if necessary with POKES
30 REM Now set machine code variables
32 INPUT "Starting address of code";S
34 INPUT "Address of final byte + 1";F
36 LET L=F-S : POKE 32769,S-(INT(S/256))
38 POKE 32770,INT(S/256)
40 POKE 32772,L-(INT(L/256))
42 POKE 32773,INT(L/256)
46 PRINT "PRINT USR 23296 on Spectrum"
48 PRINT "Computers may be linked now"
50 PRINT "Press key "1" on the Spectrum"
55 PRINT "then press "2" on host computer"
60 INPUT X: IF X<>2 GOTO
60 70 PRINT "Checksum value is ";
80 PRINT USR(X):REM or whatever syntax needed
   to invoke machine code program
90 PRINT (PEEK 32775)+((PEEK 32775)*256)
95 STOP
```

Listing 1. The BASIC program for the host computer.

```
006D                ;====================================
006D                ;THE LEECH (RECEIVE)
006D                ;====================================
006D
006D                ;SPECTRUM CODE
006D                ;AT ANY TIME DURING TRANSFER
006D                ;SPECTRUM ZERO KEY ABORTS
006D
5B00                         ORG 23296   ;IN THE PRINT BUFFER
5B00
5CB0                NMIADD   EQU 23728
FEDF                P_A_C    EQU OFEDFH ; 11111110 11011111 65247
FFDF                P_B_C    EQU OFFDFH ; 11111111 11011111 65503
FCDF                P_A_D    EQU OFCDFH ; 11111100 11011111 64735
FDDF                P_B_D    EQU OFDDFH ; 11111101 11011111 64991
FDDF
5B00 21B05C         SS       LD HL,5CB0H    ;Safe default value!
5B03 010100                  LD BC,1        ;Safe default value!
5B06 C5                      PUSH BC        ;HL contains start address and
5B07 CD655B                  CALL PIO1      ;BC contains the length
5B0A CD785B                  CALL PIO2
5B0D C1                      POP BC
5B0D
5B0D                ;WAIT FOR KEYPRESS "1"
5B0D
5B0E AF                      XOR A
5B0F 32085C                  LD (23560),A
5B12 3A085C         KBD      LD A,(23560)
5B15 FE31                    CP 31H
5B17 20F9                    JR NZ KBD
5B17
5B17                ;RECEIVING NOW   KEY "0" ABORTS
5B17
5B19 1600                    LD D,0
5B1B DD210000                LD IX,0        ;Zero checksum counter
5B1F F3                      DI             ;Disable interrupts
5B1F
5B20 C5             M_LOOP   PUSH BC        ;Save length
5B21 3E02                    LD A,2         ;Signal Spectrum not busy
5B23 01DFFC                  LD BC,P_A_D
5B26 ED79                    OUT (C),A
5B26
5B28 0620                    LD B,20H       ;Delay
5B2A 10FE           WWT      DJNZ WWT
5B2A
5B2C 01DFFC         LP       LD BC,P_A_D    ;Look for strobe
5B2F ED78                    IN A,(C)
5B31 CB67                    BIT 4,A        ;Wait until strobe detected or
5B33 280D                    JR Z FOUND
5B35 01FEEF                  LD BC,61438
5B38 ED78                    IN A,(C)       ;until an abort detected
5B3A F6E0                    OR 0E0H        ;11100000
5B3C FEFF                    CP 0FFH
5B3E 2022                    JR NZ ABORT
5B40 18EA                    JR LP
5B40
5B42 01DFFC         FOUND    LD BC,P_A_D
5B45 3E22                    LD A,22H
5B47 ED79                    OUT (C),A      ;Signal Spectrum busy
5B47
5B49 01DFFD                  LD BC,P_B_D
5B4C ED78                    IN A,(C)       ;Read data and
5B4E 77                      LD (HL),A      ;load into memory
5B4F 5F                      LD E,A
5B50 DD19                    ADD IX,DE      ;Adjust checksum
5B52 23                      INC HL         ;Advance memory pointer
5B53 C1                      POP BC         ;Recover and
5B54 0B                      DEC BC         ;adjust length
5B55 78                      LD A,B
5B56 B1                      OR C
5B57 20C7                    JR NZ M_LOOP
5B57
5B59 DDE5           EXIT     PUSH IX        ;Get checksum
5B5B C1                      POP BC         ;into BC
5B5C ED43B05C                LD (NMIADD),BC;and into NMIADD
5B60 FB                      EI             ;Enable interrupts
5B61 C9                      RET
5B62 C1             ABORT    POP BC         ;Adjust stack
5B63 18F4                    JR EXIT
5B63
5B65 3ECF           PIO1     LD A,0CFH
5B67 01DFFE                  LD BC,P_A_C
5B6A ED79                    OUT (C),A      ;Set A to control mode
5B6A
5B6C 3ED1                    LD A,0D1H      ;xx01xxxx Bit4 input, Bit5 output
5B6E ED79                    OUT (C),A
5B6E
5B70 3E22                    LD A,22H       ;Set Bit5 high
5B72 01DFFC                  LD BC,P_A_D
5B75 ED79                    OUT (C),A
5B77 C9                      RET
5B77
5B78 01DFFF         PIO2     LD BC,P_B_C
5B7B 3E4F                    LD A,4FH       ;Set B to input mode
5B7D ED79                    OUT (C),A
5B7F C9                      RET
5B7F
5B80 01DFFF         PIO3     LD BC,P_B_C
5B83 3E0F                    LD A,0FH       ;Set B to output mode
5B85 ED79                    OUT (C),A
5B87 C9                      RET
5B87
5B88                THEND    EQU $
0088                NOBYT    EQU THEND-SS
```

Listing 2. The Leech Receive program (for the Spectrum).

computer's USR may be interpreted as a signed integer and will therefore be different from the unsigned integer returned by the PEEKs. The Spectrum USR returns an unsigned integer.

LEECH LISTING 2

The code starts at 5B00 - 5B05 hex (23296 - 23301 decimal) by loading default values for the starting address (into HL) and the length counter (into BC). You should change these default values to the ones you require by POKEing the values into the program using

```
LET A=xxxxx : POKE 23297,
A-(INT(A/256)) : POKE
23298,INT(A/256)
```

for the starting address and

```
LET A=xxxxx : POKE 23300,
A-(INT(A/256)) : POKE
23301,INT(A/256)
```

for the length counter. If you do not wish to use the latter, set it to a high value, say 64000. The transfer process will normally stop when the host computer fails to send out strobe pulses, but if something went wrong and the strobe continued *ad infinitum* then the BC counter will halt the receiving program. This would prevent any precious code in Spectrum high memory being scrambled by an errant transmit program! The location NMIADD receives the checksum, but if you run the program by PRINT USR 23296 the value will be printed on the screen anyway.

LEECH LISTING 3

This listing must be entered carefully, using an assembler if you have one. Attention must be given to the EQUates so that they match your system parameters. P_DATA takes the value representing the

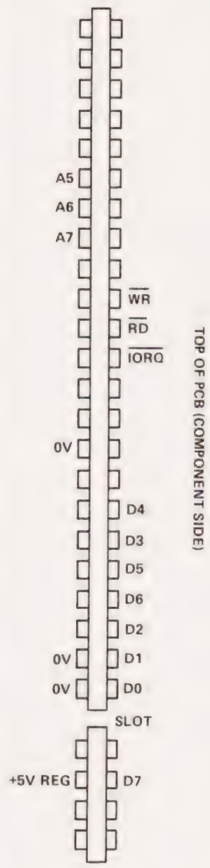


Fig. 3 Pinout of the Spectrum bus (as it appears looking into the back of the computer).

```
00DC                 ;==================================
00DC                 ;THE LEECH  (TRANSMIT)
00DC                 ;    CODE FOR THE HOST COMPUTER
00DC                 ;==================================
00DC
0005                 P_DATA    EQU 5
0004                 P_C_STRB  EQU 4
0004                 P_C_BUSY  EQU 4
0006                 PACON     EQU 6
0007                 PBCON     EQU 7
0007
0000                 BUSBIT    EQU 0
0001                 STRB      EQU 1
0001
8000                           ORG 8000H
8000                           ;8000H = 32768
8000
8000 210000          USR       LD HL,0        ;32769 Load up start address
8003 010000                    LD BC,0        ;32772 Load up length
8006 11                        DB 11H         ;Essential dummy byte
8007 0000            CHECK     DW 0           ;32775 Checksum store
8009 000000                    DB 0,0,0       ;Call PIO if necessary
800C F3                        DI             ;DI if desirable
800C
800D 1600                      LD D,0
800F DD210000                  LD IX,0        ;Zero checksum counter
800F
8013 C5              LOOP      PUSH BC        ;Preserve length
8014 7E                        LD A,(HL)      ;Read a byte
8015 5F                        LD E,A         ;and
8016 DD19                      ADD IX,DE      ;add it into checksum
8016
8018 F5                        PUSH AF
8019 010400                    LD BC,P_C_BUSY
801C ED78            WT        IN A,(C)       ;Wait until Spectrum is
801E CB47                      BIT BUSBIT,A   ;not busy
8020 20FA                      JR NZ WT       ;Memory-mapped systems
8022 F1                        POP AF         ;change ED78 to 0A00
8022
8023 010500                    LD BC,P_DATA
8026 ED79                      OUT (C),A      ;Transmit the byte
8028 CB8F                      RES STRB,A     ;Prepare the strobe bit
802A 010400                    LD BC,P_C_STRB
802D ED79                      OUT (C),A      ;Transmit the strobe
802D                           ;DELAY         Memory-mapped systems
802F 0610                      LD B,10H       ;change ED79 to 0200
8031 10FE            WTT       DJNZ WTT       ;on 3 occurances
8033 010400                    LD BC,P_C_STRB
8036 CBCF                      SET STRB,A     ;Prepare strobe off
8038 ED79                      OUT (C),A      ;Transmit strobe off
8038
803A 23                        INC HL         ;Advance memory pointer
803B C1                        POP BC         ;Recover length
803C 0B                        DEC BC         ;Count down
803D 78                        LD A,B         ;and see if all
803E B1                        OR C           ;bytes have been sent
803F 2805                      JR Z DONE
8041 000000                    DB 0,0,0
8044 18CD                      JR LOOP
8044
8046 DDE5            DONE      PUSH IX        ;Put checksum
8048 C1                        POP BC         ;into BC and
8049 ED430780                  LD (CHECK),BC  ;into store
804D 00                        NOP            ;Move BC into AB
804E 00                        NOP            ;for some BASICs
804F FB                        EI
8050 C9                        RET
8050
8050                 ;EXAMPLE OF PIO SETTING UP PROCEEDURE
8050                 ;WHICH MIGHT BE REQUIRED ON A HOST COMPUTER
8050                 ;THE PIO IS THE Z80PIO - MK3881
8050                 ;
8051 C5              PIO       PUSH BC
8052 3ECF                      LD A,0CFH      ;Make A lines control mode
8054 010600                    LD BC,PACON
8057 ED79                      OUT (C),A      ;OR use 0200
8059 3EFD                      LD A,0FDH      ;xxxxxx01 Bit0 input, Bit1 output
805B ED79                      OUT (C),A      ;OR use 0200
805B
805D 3E0F                      LD A,0FH       ;Make B lines output mode
805F 010700                    LD BC,PBCON
8062 ED79                      OUT (C),A      ;OR use 0200
8062
8064 CBCF                      SET STRB,A     ;Transmit strobe off (high)
8066 010400                    LD BC,P_C_STRB
8069 ED79                      OUT (C),A      ;OR use 0200
806B C1                        POP BC
806C C9                        RET
806C
806D                 FIN       EQU $
006D                 BYTES     EQU FIN-USR
```

Listing 3. The Leech Transmit program (for the host computer).

```
0088            ;=====================================
0088            ;                     A PRINTER DRIVER
0088            ;=====================================
0088            ;0054H=00084  bytes long
0088            ;LLIST & LPRINT
0088
5C4F            CHANS     EQU 23631
5C4F
5B88 2A4F5C     INIT      LD HL,(CHANS) ;Get CHANS
5B8B 110F00               LD DE,15      ;address and
5B8E 19                   ADD HL,DE     ;calculate
5B8F 11965B               LD DE,PATCH   ;printer channel
5B92 73                   LD (HL),E     ;and load
5B93 23                   INC HL        ;it with the
5B94 72                   LD (HL),D     ;new routine
5B95 C9                   RET
5B95
5B96 FEA5       PATCH     CP 0A5H       ;Test for token
5B98 303E                 JR NC L17
5B9A FE0D                 CP 0DH        ;Test for CR
5B9C 2809                 JR Z L12
5B9E FE20                 CP 20H        ;Test codes 0-19H
5BA0 D8                   RET C
5BA1 FE80                 CP 80H        ;Test for UDG & Block Graphics
5BA3 3802                 JR C L12
5BA5 3E20                 LD A,20H      ;Change graphics to a space
5BA7 CDB35B     L12       CALL PRINT    ;Print a character
5BAA FE0D                 CP 0DH        ;If your printer gives an automatic
5BAC C0                   RET NZ        ;linefeed change RETNZ to RET (C9)
5BAD 3E0A                 LD A,0AH
5BAF CDB35B               CALL PRINT    ;Print a linefeed if needed
5BB2 C9                   RET
5BB2
5BB3 CD655B     PRINT     CALL PIO1     ;Set PIO in case
5BB6 CD805B               CALL PIO3     ;i$ has been meddled with
5BB9 F5                   PUSH AF       ;Preserve character to be printed
5BBA F5                   PUSH AF       ;Local save
5BBB 01DFFC               LD BC,P_A_D
5BBE ED78       RDY       IN A,(C)
5BC0 CB67                 BIT 4,A
5BC2 20FA                 JR NZ RDY     ;Wait until printer not busy
5BC4 F1                   POP AF
5BC5 01DFFD               LD BC,P_B_D
5BC8 ED79                 OUT (C),A     ;Send character
5BCA 01DFFC               LD BC,P_A_D
5BCD AF                   XOR A
5BCE ED79                 OUT (C),A     ;Send strobe
5BD0 F5                   PUSH AF       ;Short delay
5BD1 F1                   POP AF
5BD2 CBE7                 SET 4,A
5BD4 ED79                 OUT (C),A     ;Cancel strobe
5BD6 F1                   POP AF
5BD7 C9                   RET
5BD7
5BD8 CD520B     L17       CALL 0B52H    ;Use ZX ROM to print
5BDB C9                   RET           ;the reserved word
5BDB
5BDC            FINN      EQU $
0054            PRLEN     EQU FINN-INIT
00DC            TOTLEN    EQU FINN-SS
```

Listing 4. A printer driver.

address of the port putting out the parallel DATA and similarly P_C_STRB the port number for the strobe signal, P_C_BUSY the port number for the busy (or acknowledge) signal. If you are unlucky, the host system might have two more port addresses associated with the control function of the Z80 PIO chip itself. These are PACON and PBCON for the control of the A lines and B lines respectively, but I would suggest a careful scrutiny of the wiring diagram around the PIO to discover which lines are being used for which function.

The final equates to change will be BUSBIT (the bit-number of the BUSY/ACKNOWLEDGE signal) and STROBE (the bit-number of the STROBE signal). If your host computer has ports which are memory-mapped you will need to change all IN A,(C) instructions to LD A,(BC) and all OUT (C),A instructions to LD (BC),A. The alternative code is given in the comments of the source listing.

Both the transmit and receive listings are hopefully explicit enough to help users who come across strange Centronics hardware. I am thinking in particular of the standard RML 380Z which delivers an antiquated seven bits of data with BIT7 carrying the strobe (have fun with that one!) or the slightly more eccentric Memotech MTX500 which puts the Strobe OUT using an IN command and cancels the Strobe automatically when reading the Busy line . . .

LEECH LISTING 4

Finally there is the bonus of Listing 4. If you decide to use it you *must* disconnect the Leech Link, not only because you will probably need a ribbon cable connection but also because the Leech PIO is reconfigured to give an output which is intended for a printer and not for the host computer! The routine itself is a 'no frills' affair, trimmed down to fit the printer buffer. The buffer ends at 5BFF hex so there is only some 20 (hex) bytes spare, which isn't quite enough for a screen dump routine.

INTER RUPTS

Richard Sargent

Interrupting your microprocessor is a very powerful tool, rather than a sign of ill-breeding. In this article we look at the non-maskable interrupt of the Z80 CPU and the way in which it may be put to use.

An interrupt is a signal from the outside world which causes the CPU to stop its current activity and to execute another program before returning to the initial program and continuing with it. The Z80 CPU has two interrupt inputs, known as the non-maskable interrupt (NMI) and the maskable interrupt (INT).

MASKABLE INTERRUPT

The Z80 can be programmed to respond to this interrupt in any one of three possible modes. However, since two of the three modes require a peripheral device to supply the interrupt and a data byte, and the third one causes a jump to 0038 hex where the NAS-SYS delay routine resides, the maskable interrupt does not lend itself to simple experimentation.

NON-MASKABLE INTERRUPT

The non-maskable interrupt is far easier to use and it is an interrupt which will be accepted at all times by the CPU. The NMI is invoked by taking pin 17 of the Z80 chip to ground. A pulse on this pin sets an internal NMI latch which is tested by the Z80 at the end of every instruction. If the latch is set, the CPU responds by pushing the program counter onto the stack, and jumping to location 0066 hex. The service routine for the non-maskable interrupt must therefore begin at this location.

In the Nascom this location is in ROM and it contains another jump, this time to 0C7D hex, which is in RAM. 0C7D hex, labelled $NMI, normally holds the final jump, this time to 0475 hex, which takes us back into ROM and into the actual interrupt routine, which drives the monitor's single-step mechanism. $NMI is known as a 'jump vector', and its contents can be changed at will by the programmer. In this way the single-step mechanism is replaced by a different routine, which can be any program the user likes to write, lengthy or short, complex or simple. The CPU will run it so long as a few basic rules are observed.

THE PROGRAM

The demonstration progam offered is intended only as an introduction to the non-maskable interrupt, and the relevant device manuals should be consulted for further information, particularly on the maskable interrupt modes, which are extremely powerful tools.

The program is written for the NAS-SYS 3 monitor, which can be interrupted safely. Unfortunately the NAS-SYS 1 monitor has a bug in it which causes data which is needed later in the program to be left unprotected by the stack pointer. If an interrupt were to occur at this time, the action of stacking the program counter would destroy two bytes of data, resulting in a crash shortly after the interrupt routine had passed control back to the main routine. How often an interrupt would occur at this critical time is hard to estimate, and NAS-SYS 1 owners are therefore encouraged to 'carry on regardless'. To run under NAS-SYS 1, it is necessary to change a few bytes of the interrupt program, as shown in Table 1.

To experiment with the program, a small hardware installation is required; a push-button switch is wired onto pin 6 of the 77-way Nasbus, and closure of the switch takes pin 6 to ground.

OPERATION OF THE NMI PROGRAM

To initialise the new NMI facility, Execute A000 A500 nnnn. A500 is the start of the initial program which is to be run; in this case it is a small demonstration display on the VDU. nnnn is optional and represents the start address of an alternate program which can be run while the initial program is on interrupt. It should be noted however that this alternate program must end with a normal RETurn (code C9).

TABLE 1

LOCATION	NEW CODE
A018	21 40 02
A026	11 88 A4
A0A0	CD B5 02
A13F	CD B5 02
A144	C3 8C A4

The features of the interrupt program itself are:

- Freeze the screen display, save it on tape, print it.
- Examine or Modify memory.
- Use NAS-SYS commands.
- Display the Z80 registers (NAS-SYS 3 only).
- Restore the screen display (or load in a new one) and continue.
- Execute an alternate program, then return to the main program.

When the interrupt button is pressed, the interrupt program saves all registers, the NAS-SYS workspace and the screen display. The screen display is now frozen, since the initial program has been put aside. The keyboard is being scanned and two options are available to the user.

- S for "skip" ends the interrupt and passes control back to the initial program, with the contents of the program and the screen unaltered.
- Any other key-press clears the screen and brings up the prompt "— — N M I — —", whereupon the interrupt program again scans the keyboard.

THE SECOND OPTION

NAS-SYS COMMANDS

NAS-SYS commands can now be used, but $NMI must not be reset to its original value of 0475, otherwise when the next interrupt is received the monitor's

Listing 1. Source code for the Interrupt program.

```
A52A            ;GENERAL PURPOSE INTERRUPT PROGRAM
A52A            ;R SARGENT  APRIL 1982
A52A            ;V2
A52A            ;
A52A                      ;
02F0            INLIN     EQU 02F0H;in nas-sys 3 (else 02B5H)
0280            MODIFY    EQU 0280H;in nas-sys 3 (else 0240H)
0068            MODLEN    EQU 0068H;length of "modify"
0012            MODOFF    EQU 18;for nas-sys 3   (else 16)
0C0A            ARGC      EQU 0C0AH
0C0B            ARGN      EQU 0C0BH
0C7D            $NMI      EQU 0C7DH
0C77            $UOUT     EQU 0C77H
0C77            PRINTR    EQU $UOUT;or addr of printer subroutine
0C77
0C77            ;NAS-SYS DF CALLS
0C77
00DF            DF        EQU 0DFH
0079            RLIN      EQU 79H
0060            ARGS      EQU 60H
005C            SCALJ     EQU 5CH
006B            ERRM      EQU 6BH
006B                      ;
006B                      ;
A000                      ORG 0A000H
A000                      LOAD 0A000H
A000
A000                      ;E A000 A500 nnnn where A500 is the
A000                      ;start of the initial program and
A000                      ;n is the start of a second program
A000                      ;which is optional
```

```
A000                            ;
A000 2130A0     INIT            LD HL,START
A003 227E0C                     LD ($NMI+1),HL
A006 DF60                       DB 0DFH,60H
A008 3A0B0C                     LD A,(ARGN)
A00B FE03                       CP 3          ;3 ARGS ENTERED?
A00D 3003                       JR NC OK1     ;NOT UNDER 3 SO OK
A00F 012BA1                     LD BC,DEFAL   ;DEFAULT INTO BC
A012 ED4329A1 OK1               LD (NWPROG),BC
A016 EB                         EX DE,HL
A016
A017 E5                         PUSH HL
A017                            ;move modify routine
A017                            ;out from monitor
A018 218002                     LD HL,MODIFY
A01B 1178A4                     LD DE,MODRAM
A01E 016800                     LD BC,MODLEN
A021 EDB0                       LDIR
A021                            ;patch modify
A023 2147A1                     LD HL,PSTR
A026 118AA4                     LD DE,MODRAM+MODOFF
A029 010400                     LD BC,4
A02C EDB0                       LDIR
A02C
A02E E1                         POP HL
A02F E9                         JP (HL);run the initial program
A02F                            ;
A02F                            ;
A02F                            ;
A02F            ;Come HERE after NMI
A02F            ;
A030 D5         START           PUSH DE
A031 DDE5                       PUSH IX
A033 F5                         PUSH AF
A033                            ;
A034 C5                         PUSH BC
A035 FDE5                       PUSH IY
A037 E5                         PUSH HL
A037                            ;
A037                            ;
A037                            ;
A037                            ;save workspace at 0C00-0C2B inc
A037                            ;
A03B 114CA1     SETUP           LD DE,SAVC00
A03B 21000C                     LD HL,0C00H
A03E 012C00                     LD BC,2CH
A041 EDB0                       LDIR
A041                            ;
A041                            ;
A041                            ;screen to scratchram
A041                            ;
A043 AF                         XOR A
A044 324BA1                     LD (LATCH),A
A047 CD4CA0                     CALL MOVE
A04A 1834                       JR PROG
A04A                            ;
A04A                            ;
A04C 2148A1     MOVE            LD HL,SVDU-48
A04F FD21CA07                   LD IY,080AH-64;IY points to vdu-64
A053 0610                       LD B,16
A055 C5         LP              PUSH BC
A056 113000                     LD DE,48
A059 19                         ADD HL,DE     ;HL points into scratch
A05A 114000                     LD DE,64
A05D FD19                       ADD IY,DE     ;IY points into vdu
A05F FDE5                       PUSH IY
A061 D1                         POP DE        ;DE points into vdu
A062 3A4BA1                     LD A,(LATCH)
A065 B7                         OR A
A066 2001                       JR NZ SKIP1
A068 EB                         EX DE,HL
A069 013000     SKIP1           LD BC,48
A06C D5                         PUSH DE
A06D E5                         PUSH HL
A06E EDB0                       LDIR
A070 E1                         POP HL
A071 D1                         POP DE
A072 3A4BA1                     LD A,(LATCH)
A075 B7                         OR A
A076 2001                       JR NZ SKIP2
A078 EB                         EX DE,HL
A079 D5         SKIP2           PUSH DE
A07A FDE1                       POP IY
A07C C1                         POP BC
A07D 10D6                       DJNZ LP
A07F C9                         RET
A07F                            ;
A07F                            ;
A07F                            ;
A080 2130A0     PROG            LD HL,START
A083 227E0C                     LD ($NMI+1),HL
A083                            ;
A086 CF                         RST 8
A087 FE53                       CP "S";SKIP
A089 2879                       JR Z WINDUP
A08B FE73                       CP "s"
A08D 2875                       JR Z WINDUP
A08D                            ;
A08F EF0C00                     DB 0EFH,0CH,0
A092 EF2D2D20
A096 4E204D20
A09A 49202D2D
A09E 0D00       MLOOP           DB 0EFH,"-- N M I --",0DH,0
A09E                            ;
A0A0 CDF002                     CALL INLIN
A0A0
A0A0                            ;INLIN blinks cursor,accepts
A0A0                            ;keypresses,waits for CR,
A0A0                            ;sets DE to start of the input
A0A0                            ;line, restores HL, and RETurns
A0A0
A0A3 1A                         LD A,(DE)
A0A3
A0A3                            ;Recover command letter
A0A3
A0A4 FE41                       CP "A"
A0A6 2826                       JR Z OK
A0A8 FE43                       CP "C"
A0AA 2822                       JR Z OK
A0AC FE45                       CP "E"
A0AE CA21A1                     JP Z EXEC
A0B1 FE47                       CP "G"
A0B3 2819                       JR Z OK
A0B5 FE49                       CP "I"
A0B7 2815                       JR Z OK
A0B9 FE4D                       CP "M"
A0BB CA2DA1                     JP Z MOD
A0BE FE53                       CP "S"
A0C0 2842                       JR Z WINDUP
A0C2 FE10                       CP 10H;control P for printout
A0C4 281A                       JR Z PRINT
A0C6 FE4B                       CP "K"
A0C8 380C                       JR C PERR
A0CA FE59                       CP "X"+1
A0CC 3008                       JR NC PERR
A0CC                            ;
A0CE 320A0C     OK              LD (ARGC),A ;store command letter
A0D1 13                         INC DE
A0D2 DF79                       DB DF,RLIN ;convert the line
A0D4 3004                       JR NC PEND
A0D4
A0D6 DF6B       PERR            DB DF,ERRM
A0D8 18B8                       JR MLOOP
A0D8
A0DA DF60       PEND            DB DF,ARGS ;args into HL,DE,BC
A0DC DF5C                       DB DF,SCALJ;call routine held in ARGC
A0DE 18B2                       JR MLOOP
A0DE                            ;
A0DE                            ;
A0DE                            ;
A0E0 2147A4     PRINT           LD HL,SVDUT-1
A0E3 0E01                       LD C,1
A0E5 CDF2A0                     CALL PR2
A0E8 2177A1                     LD HL,SVDU-1
A0EB 0E0F                       LD C,15
A0ED CDF2A0                     CALL PR2
A0F0 18A0                       JR MLOOP
A0F0                            ;
A0F0                            ;
A0F0                            ;
A0F2 0630       PR2             LD B,48
A0F4 23         PR3             INC HL
A0F5 7E                         LD A,(HL)
A0F6 CD770C                     CALL PRINTR
A0F9 10F9                       DJNZ PR3
A0FB 3E0D                       LD A,0DH      ;load a CR/LF
A0FD CD770C                     CALL PRINTR
A100 0D                         DEC C
A101 20EF                       JR NZ PR2
A103 C9                         RET
A103                            ;
A103                            ;------prepare to exit-----
A103                            ;
A103                            ;
A103                            ;restore screen
A103                            ;
A104 3EFF       WINDUP          LD A,0FFH
A106 324BA1                     LD (LATCH),A
A109 CD4CA0                     CALL MOVE
A109                            ;
A109                            ;
A109                            ;restore nas-sys workspace
A109                            ;
A10C 214CA1                     LD HL,SAVC00
A10F 11000C                     LD DE,0C00H
A112 012C00                     LD BC,2CH
A115 EDB0                       LDIR
A115                            ;
A115                            ;
A117 E1                         POP HL
A118 FDE1                       POP IY
A11A C1                         POP BC
A11A                            ;
A11B F1                         POP AF
A11C DDE1                       POP IX
A11E D1                         POP DE
A11E                            ;
A11F ED45                       RETN
A11F                            ;
A121 2104A1     EXEC            LD HL,WINDUP
A124 E5                         PUSH HL
A125 2A29A1                     LD HL,(NWPROG)
A128 E9                         JP (HL)
A128                            ;
A129            NWPROG          DS 2
A12B C9         DEFAL           RET
A12C 00                         NOP
A12C                            ;
A12C
A12D            PRGFN1          EQU $
A12D
A12D            ;The code from here until PRGFN2
A12D            ;deals with the MODIFY command
A12D
A12D 13         MOD             INC DE
A12E DF79                       DB DF,RLIN ;convert the line
A130 3005                       JR NC MOD2
A132 DF6B                       DB DF,ERRM
A134 C392A0                     JP MLOOP
A137 DF60       MOD2            DB DF,ARGS ;args into HL,DE,BC
A139 CD78A4                     CALL MODRAM;call the new "modify"
A13C C392A0                     JP MLOOP
A13C                            ;
A13F CDF002     PATCH           CALL INLIN
```

```
A142 DF64                  DB 0DFH,64H
A144 C38EA4                JP MODRAM+MODOFF+4
A144                       ;
A147 C3        PSTR        DB 0C3H
A148 3FA1                  DW PATCH
A14A 00                    DB 0
A14A
A14A
A14B           PRGFN2      EQU $
A14B                       ;
A14B                       ;
A14B           LATCH       DS 1
A14C           SAVCOO      DS 2CH
A178           SVDU        DS 15*48
A448           SVDUT       DS 48
A448
A478           SCRFN1      EQU $
A478
A478           MODRAM      DS 68H
A478                       ;
A4E0           SCRFN2      EQU $
A4E0                       ;
A4E0                       ;
A500                       ORG 0A500H
A500                       LOAD 0A500H
A500           ;
A500           ;Initial program for test purposes
A500                       ;
A500 EF0C00                DB 0EFH,0CH,0
A503 3E31      AGAIN       LD A,31H
A505 32CA0B                LD (0BCAH),A
A508 3C                    INC A
A509 32F90B                LD (0BF9H),A
A50C 3C                    INC A
A50D 320A08                LD (080AH),A
A510 3C                    INC A
A511 323908                LD (0839H),A
A514 3C                    INC A
A515 328A0B                LD (0B8AH),A
A518 3C                    INC A
A519 32B90B                LD (0BB9H),A
A51C ED5F      L4          LD A,R
A51E CBFF                  SET 7,A
A520 32A109                LD (09A1H),A
A523 F5                    PUSH AF
A524 3EFF                  LD A,0FFH
A526 FF                    RST 38H
A527 F1                    POP AF
A528 18F2                  JR L4
A528                       ;
A52A           TSTEND      EQU $
A52A
A52A
3ED4 4074 A500 A52A A52A 00
```

standard display of registers and entry into single step mode of operation would take place.

Whenever a NAS-SYS command is used, control normally passes through a subroutine called INLS which unhappily resets the $NMI. This routine is therefore partially by-passed in the NMI program. In their own right, most NAS-SYS commands do not reset $NMI and so it is quite possible to TABULATE, QUERY or OUTPUT to ports, COPY data, display program registers (P command in NAS-SYS3), and WRITE to and READ from cassette tape. The last two commands give the facility, should it be needed, to manipulate the data on the screen.

MODIFYING THE SCREEN

The display is saved on tape by use of the command W A178 A478. A178 (label SVDU) marks the start of the display-save area and A478 (label SCRFIN) marks the end of it. The Read command can load this area of RAM just like any other and so it is possible to over-write the saved display with another display that had been stored on tape from a previous occasion.

THE MODIFY COMMAND

MODIFY is a NAS-SYS routine which does reset $NMI whenever it is used, so this is one routine that has been modified so that it works successfully after every interrupt.

PRINTING THE SCREEN

PRINT (control P) sends the contents of the screen at the time of interrupt to the user's printer. It expects a printer subroutine to exist at 0E00 hex. The CALLs to the printer are at A0F6 hex and A0FD hex and can be changed as necessary. For example, DF 6F 00 instead of CD 00 0E would send the data to the Nascom's serial port.

THE EXECUTE COMMAND

EXECUTE has been re-written, and its function is to execute the alternate program, if a starting address for the alternate program has been specified. If no such address is available, then 'Execute' operates in the same manner as 'Skip'.

As to the nature of this alternative program, that is left (as is usual in such cases), to the ingenuity of the reader . . .

BOOK PAGE

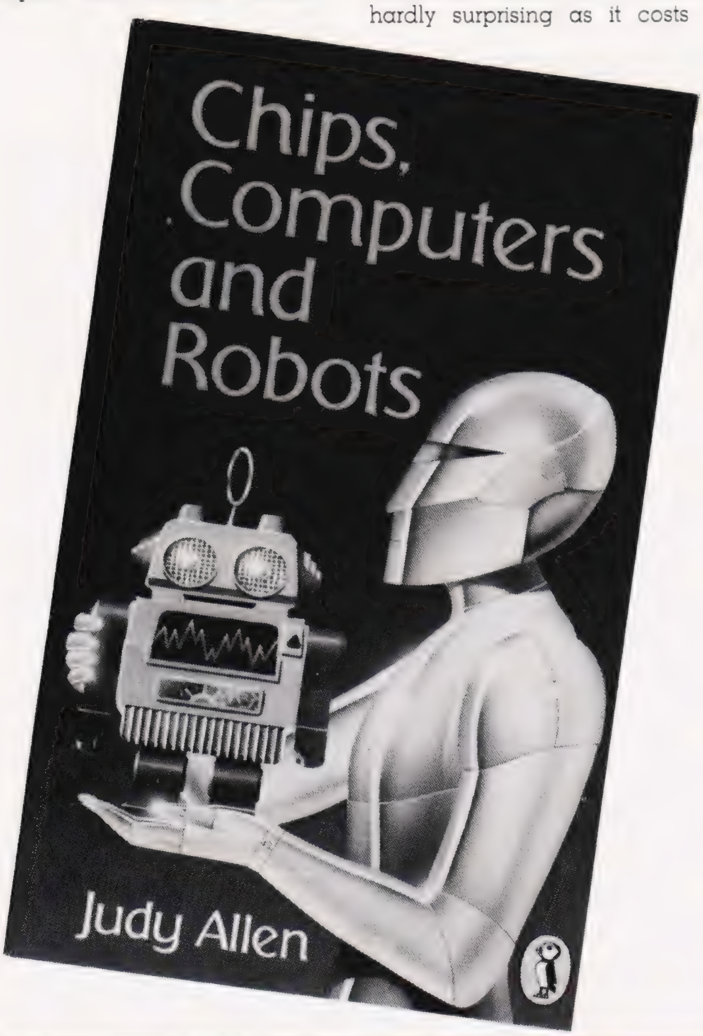
Garry Mashall

Our reviewer takes a look at two books about robots, and some reference works on a variety of topics.

It seems that after personal computers, the next thing is to be personal robots. With this in mind, I have been on the lookout for books about robots, and this month two have turned up with references to them in their titles. To be clear, neither of them is about personal robots specifically, although both make mention of personal robots and the problems associated with their design in their closing pages. However, these books can provide us with an introduction to robots and robotics, as well as giving us some appreciation of what a personal robot might be like and what it could do for us.

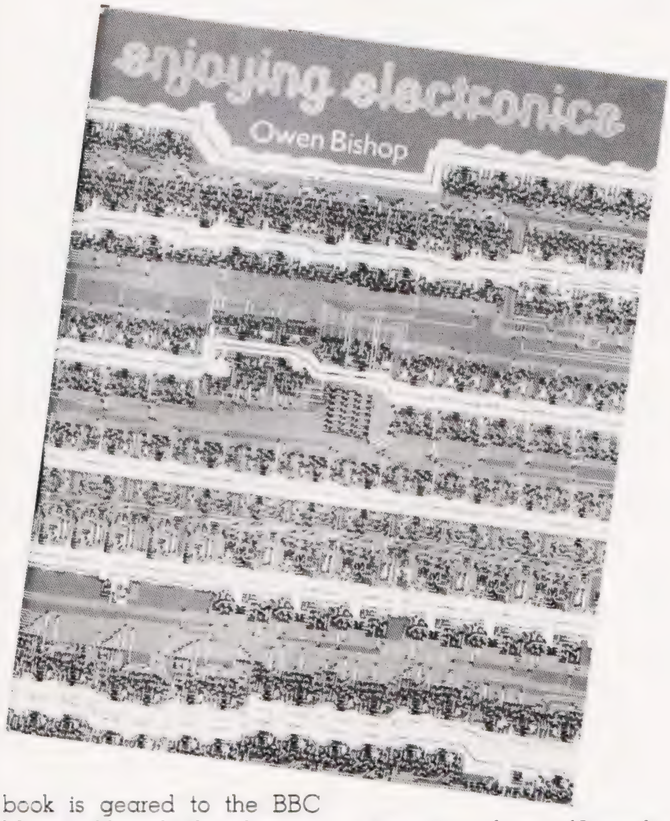
Chips, Computers and Robots by Judy Allen is a Puffin book, and its presentation is at the introductory level that this might lead you to expect. It contains 32 short chapters, each of two or three pages. Starting with the preparation and manufacture of microelectronic circuits, it proceeds to computers and how they work, takes an unscheduled detour into Information Technology to cover topics that include satellite communications and Teletext, before reaching robots and robotics. This is an awful lot of ground to cover in 80 brief pages, and the treatment is necessarily selective. The format of the book does not allow it to bring out at all well how chips make up computers or how computers contribute to robotics.

As a non-technical introduction to an area that is now covered by a number of books it is adequate, although occasional instances of lack of precision and of attempts at simplification that give rise to misleading statements are irritating. The illustrations are disappointing. They are poorly presented, particularly by comparison with the hardback edition, but this is hardly surprising as it costs about five times as much. More important, perhaps, is that, particularly in the first half of the book, they are often of dubious relevance to the text.



The last 10 chapters cover robotics, and are much the best part of the book. Following the apparently obligatory acknowledgement of the robot's origins in fiction, there are chapters on exoskeletons (which can be 'worn' to amplify enormously human muscle power), remote manipulators (as used in hazardous environments and for bomb disposal), and industrial robots. The 'housework' robot is then introduced. The difficulty in producing a robot to be useful in the home is that housework is actually very varied and complex, and totally unlike the kind of repetitive work that almost all industrial robots currently carry out. I suppose that a personal robot could be expected to hoover, mow the lawn, make the beds and fetch a beer from the fridge when you wanted one, among other things, although I must admit to some scepticism about the need for one. The book's final chapter on artificial intelligence is not very penetrating, but it does point us in the right direction to face the future. I suppose that as a non-technical introduction to microelectronics, all that stems from it, and that which may develop with its aid in the future, this little book is quite acceptible.

DIY Robotics And Sensors by John Billingsley is essentially a series of projects for constructing gadgets to be controlled by a computer. The



book is geared to the BBC Micro, although the descriptions are sufficiently clear that the projects could be adapted for any other micro having comparable facilities (not that I can think of one!). Throughout the book the hardware and software aspects of the projects are presented side by side so that a typical chapter contains not only constructional and circuit diagrams but also programs, usually in BASIC but occasionally in assembly code too.

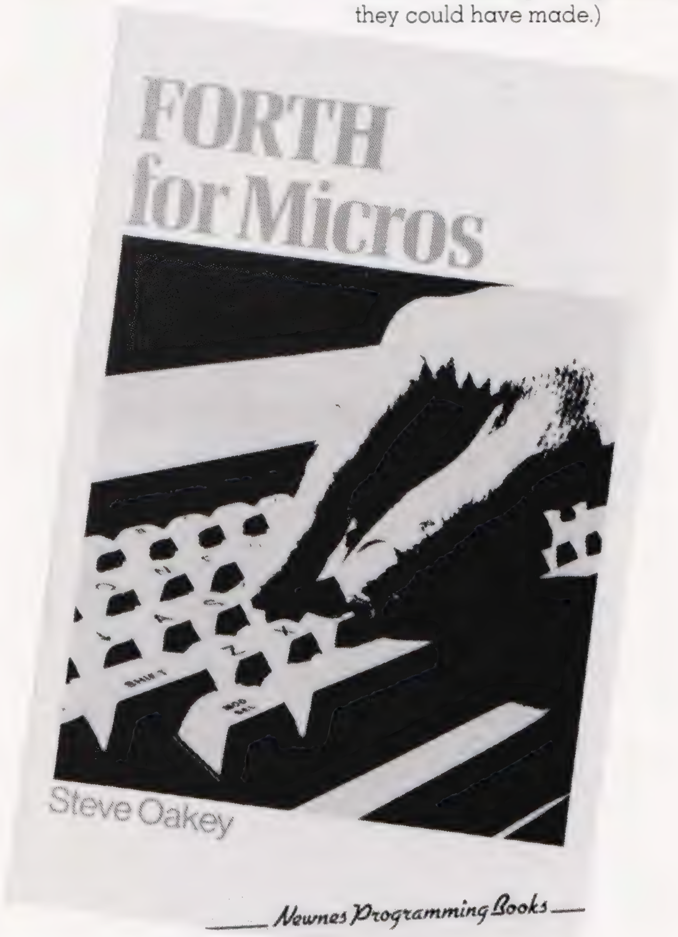
John Billingsley is a lecturer at Portsmouth Polytechnic, and is well-known as the organiser of the Euromicro 'micromouse' competition. He discusses the competion in his final chapter, suggesting that it has now served its purpose as a stimulant to developments. Because the problems involved have all been effectively solved, he is now proposing a robot ping-pong contest as the means for testing and stretching the abilities of all those interested in robotics. In case this makes his involvement in robotics seem anything less than serious, I would mention that he is involved in active research and development with the 'Craftsman robot' project at Portsmouth Polytechnic, and that a recent account of this project is presented in *Electronics and Power* for November/ December 1983.

The projects in the book cover joysticks, a light pen, stepper motors, a turtle, interfacing a robot and robot vision. A certain knowledge of electronics and some expertise, in soldering, for example, will be necessary to carry out the projects, but the electronics enthusiast will cope easily and nothing is demanded that is beyond a fairly determined beginner. In the best tradition of the English inventor, readily available materials are made use of rather than expensive special components.

The drawings in the book are an education in themselves. Reproduced (as I imagine) from the author's freehand sketches, they show that precise engineering information can be conveyed without the need for a high degree of artistic ability. In fact, as images, some of the drawings (including that of the joystick on page 26) are highly intriguing under close examination because of their lack of perspective.

Should you feel the need for a basic introduction to practical electronics before moving on to robot projects, **Enjoying Electronics** by Owen Bishop may be just what you need. Starting absolutely from square one, it presents a series of experiments to illustrate and explain what electronics is all about. We move from currents flowing through lamps to resistors, diodes, transistors, simple circuits for switching and for oscillators, and relays. There is much more besides. The illustrations (professionally produced this time) are excellent, and the beginner would learn a great deal about electronics just by reading the book, but much more by doing the experiments that it describes.

To move to a different topic, a book on FORTH has just been published which I consider to be much the best that I have read. It is **FORTH For Micros** by Steve Oakey. Before going any further I must declare an interest, for I have had some involvement in creating the series on which the book belongs, and have also written two books in the same series.



FORTH For Micros provides an accessable introduction to FORTH, shows how to write programs (properly) in FORTH, and illustrates this with many examples. I know of no other book that meets more than one of these aims successfully. The reader is taken in a steady and entertaining way from '2 2 + .' to screens and the implementation of complex data structures. The book deserves a wide readership among all those interested in FORTH.

Actually, there is a connection between FORTH and robots, for at least one manufacturer, Cyber Robotics, provides FORTH as the language for programming its robots, claiming that it is much more suitable for this purpose than BASIC. (Even though true, this is not the most convincing comparison they could have made.)

Finally, I would bring to your notice some useful reference works. The Pitman programming pocket guides are genuinely pocket-sized reference works. They are sensibly and thoughtfully produced in flip chart form. Two of the most recent ones are **Assembly Language For The Z80** and **Programming For The Apple.** The format of the guides makes them ideal for presenting the mass of detail needed for assembly code programming, and the Z80 guide does just this as well as presenting a tutorial

introduction to assembly code programming for the Z80. The Apple guide contains in handy form a great deal of the information that Apple II and IIe users could ever want, and also provides a number of useful subroutines.

This month's books are:
Chips, Computers and Robots by Judy Allen (Puffin Books), 95 pages, £0.95
DIY Robotics And Sensors by John Billingsley (Sunshine), 119 pages, £6.95
Enjoying Electronics by Owen Bishop (Cambridge Educational), 94 pages, £2.70
FORTH For Micros by Steve Oakey (Newnes), 148 pages, £5.95
Assembly Language For The Z80 by Julian Ullman (Pitman), 60 pages, £2.50
Programming For The Apple by John Gray (Pitman), 62 pages, £2.50

FASTER FORTH

Martin H. Goose

Readers of Computing Today will recall the articles on the Aculab floppy tape drive and the MMS FORTH, both for the TRS-80. Now you can make them work together.

Previous articles by Dave Peckett have introduced the Aculab floppy tape drive (1) and MMS FORTH in its cassette-based version (2), to users of the TRS-80. Users of a disc-based version of FORTH will know how much more convenient it is to have UPDATEd buffers FLUSHed to disc automatically, rather than have to go through the manual equivalent with the cassette-based version. The purpose of this article is to explain to users of the TRS-80 and Aculab floppy tape drive who use the cassette-based MMS FORTH how they can add the necessary additional FORTH words to mimic disc operation.

BIBLIOGRAPHY

Before you read the following explanation of how the modifications are made you should review the contents of Dave Peckett's two articles and the section of the Aculab user's manual on machine language programming. You will also need to refer to the MMS glossary for the usage of the words EWBLK, ERBLK, CWBLK, and CRBLK. The code given assumes 32K of memory, and MMS Cassette FORTH V1.8. All FORTH words and BASIC commands are shown in block capitals.

VECTORING IN

Blocks are written to, and read from, cassette tape using the FORTH words CWBLK and CRBLK. Fortunately these words are not executed directly but by vectored execution. This means that addresses which point to the code for these two words are kept as two variables EWBLK and ERBLK. For example, when FORTH wants to write a block to cassette it looks for the address in EWBLK and executes the routine at that address. This normally executes CWBLK.

When executed, both CWBLK and CRBLK require the address of the buffer to be used, and the block number involved, as the second on stack and top of stack respectively. The essential part of the modification is to write two additional FORTH words which expect the same values on the stack but transfer data between block buffer and floppy tape. My names for these two new words into EWBLK and ERBLK will ensure that when a block is to be written @PUT is executed, and similarly when a block is to be read @GET is executed.

THE FLOPPY FILE

Using the floppy tape system at the machine code level requires the use of a 16-byte filespec which gives the floppy tape operating system the data it needs to move tape sectors between tape and memory. This filespec must be set up first before @PUT and @GET can be used. This is done in block 3 of Listing 1. This locates the filespec from 42F0H to 42FFH, just above the first FORTH block buffer. As a floppy tape sector is 256 bytes long and a FORTH block 1024 bytes, four sectors are needed per block.

The floppy tape operating system normally uses part of low memory, from 4300H to 47FFH. This is used for file directory maintainance, as an input/output buffer and to store various other parameters. In this position it will clash with FORTH's block buffers which are located at 4300H to 46FFH, and 4700H to 4AFFH, so it must be relocated. The obvious place to put it is the top of available memory. Having done this FORTH must be denied access to this memory area. This is achieved by lines 1 and 2 of block 3.

These are the essential features of the additional code required. This, together with some other necessary words and some optional utility words, makes up the complete package given in Listing 1. Having described the main words in Listing 1 the rest of the words have the following functions.

Block 2. The words EX, EXX, LDIR, and LDDR are extra words for the ASSEMBLER. The ASSEMBLER provided by MMS is for the 8080 microprocessor. These extra words add some of the extra power of the Z80 to the existing ASSEMBLER and are used in the definition of @PUT and @GET. Once present in the ASSEMBLER vocabulary they are also available for use in FORTH words defined using CODE.

Block 3. The only word not previously described in this block is @LIST. This functions in exactly the same way as in BASIC, loading the directory from a tape and displaying it.

Blocks 4 and 5 contain the code for @PUT and @GET.

Block 6. This block contains two essential words, FLOPPY and CASSETTE, and four optional words. FLOPPY and CASSETTE redirect input/output to the floppy tape and cassette tape respectively. They operate by storing the variables EWBLK and ERBLK. @STOP stops the floppy tape drive and @START starts the drive which then continues until it reaches the splice in the tape loop, where it stops. @BLOCK and @LOAD function in the same way as BLOCK and LOAD but stop the floppy tape after each block is brought to a block buffer. This is useful if you can format tapes as described below.

Block 7. This starts with the dummy word TRANSFERWORDS so that after using blocks 7, 8 and 9 you can FORGET TRANSFERWORDS. The word C>F transfers consecutive blocks from cassette tape to floppy tape. The values which must be put on the stack before the word is executed are shown in the definition of the word using the usual FORTH notation. The words DIDDLE and TWBLK are used in the implementation of the word F>C in the next screen.

Block 8. The word F>C transfers consecutive blocks from floppy tape to cassette tape. This implementation numbers all blocks kept on floppy tape from number 1 upwards. 80 1 9 C>F would have blocks 80 to 88 inclusive on cassette to become blocks 1 to 8 inclusive on floppy tape. 1 80 9 F>C would move them back again.

Block 9. The word @CLEAR empties consecutive blocks on floppy tape. 4 9 @CLEAR would empty blocks 4 to 9 inclusive. If you wish to clear single blocks use CLEAR.

Block 1. This block contains a FORTH application which, when loaded, places a new video driver into the TRS-80 memory map. This loader is a variation of that given in an article (3) describing a low cost lower case modification for the TRS-80. It is a FORTH equivalent of the program given in my letter to the editor in the December 1982 edition of *Computing Today.*

IMPLEMENTATION

The procedure for implementing these modifications is as follows:

1. The first step is not related to FORTH at all and must be taken before FORTH is loaded into the TRS-80. The FORTH words @GET and @PUT access a predefined floppy tape file called "SCREENS/FTH", and a dummy file of this name must be created on a floppy tape. This is done from BASIC

Listing 1.

```
BLOCK : 1

( Loader for lower case driver in FORTH )
HEX
0DD 6E 3  ( LD L,(IX+3) )  0DD 66 4  ( LD H,(IX+4) )
0DA 9A 4  ( JP C,049AH  )  0DD 7E 5  ( LD A,(IX+5) )
0B7       ( OR A        )  28 1      ( JR Z,$+1    )
77        ( LD (HL),A   )  79        ( LD A,C      )
0FE 20    ( CP 20H      )  0DA 6 5   ( JP C,0506H  )
0FE 80    ( CP 80H      )  0D2 0A6 4 ( JP NC,04A6H )
0C3 79 4  ( JP 0479H    )
CLS
: MOVEIT   DO I - DUP (R C' R) LOOP ;
1E DUP 4060 + SWAP 0 MOVEIT  ( Start address left on stack)
401E !                ( Store start address in D.C.B. )
9 0 PTC " Lower case driver loaded "
FORGET MOVEIT
DECIMAL
```

```
BLOCK : 2

(       Z80 additions to MMS 8080 assembler              )
HEX ASSEMBLER DEFINITIONS
08 1BY EX            ( EX AF,AF'             )
D9 1BY EXX           ( EXX - move to alternate BC, DE, HL )
: LDIR   B0ED , ;    ( Forwards block move )
: LDDR   B8ED , ;    ( Backwards block move )

FORTH DEFINITIONS DECIMAL
```

```
BLOCK : 3

(     Code to locate FILESPEC and define @LIST              )
HEX FORTH DEFINITIONS               BB00 4B03 !
BC00 41AD !             ( Relocate floppy directory etc.    )
53 CCONSTANT FILESPEC 43 C, 52 C, 45 C, 45 C, 4E C, 53 C,
                    20 C, ( SCREENS in ASCII + a space      )
                    46 C, 54 C, 48 C, ( FTH in ASCII        )
                    0 C,                ( Drive number      )
                    BF00 ,              ( Load address      )
                    0 ,                 ( Entry address     )

' FILESPEC 42F0 10 MOVE  ( Move FILESPEC to 42F0H-42FFH    )
FORGET FILESPEC
( 42F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF         )
(     S  C  R  E  E  N  S     F  T  H 00 00 BF 00 00        )
( @LIST   as BASIC      ( -- )                     )
CODE @LIST   EXX 42F0 DE LXI 37CE CALL EXX NEXT
```

```
BLOCK : 4

(     Code for @GET                                         )

( @GET   Bring block  (TOS) to buffer  (2OS)                )

CODE @GET    4 A MVI EX EXX BC POP BC DCX C A MOV
                         RLC RLC FC ANI DE POP EX
             BEGIN  EX PSW PUSH DE PUSH A C MOV 1 B MVI
                           42F0 DE LXI 37DD CALL
                    DE POP BF00 HL LXI 100 BC LXI LDIR
                    PSW POP A INR EX A DCR   =0 END
             EXX NEXT
```

```
BLOCK : 5

(     Code for @PUT                                         )

( @PUT  Moves buffer in (2OS) to block in (TOS)             )

CODE @PUT    4 A MVI EX EXX BC POP BC DCX C A MOV RLC RLC
                       FC ANI HL POP EX
             BEGIN  EX PSW PUSH BF00 DE LXI 100 BC LXI LDIR
                    HL PUSH A C MOV 1 B MVI 42F0 DE LXI
                       37D7 CALL HL POP
                    PSW POP A INR EX A DCR =0 END
             EXX NEXT
```

```
BLOCK : 6

(     Floppy tape utility words                      ) HEX
CODE @STOP ( stops drive) A XRA F0 OUT NEXT
CODE @START ( starts drive and parks at splice)
        3 A MVI F0 OUT NEXT
                                                      DECIMAL
: FLOPPY    ( vector I/O to floppy tape)
            (') @GET     (') ERBLK     !
            (') @PUT     (') EWBLK     !   ;
: CASSETTE ( vector I/O to cassette tape)
            (') CRBLK    (') ERBLK     !
            (') CWBLK    (') EWBLK     !   ;

: @BLOCK   ( n -- buffer address) BLOCK @STOP ;
: @LOAD    ( n -- ) DUP @BLOCK DROP LOAD ;
```

Listing 2.

```
4A00  LD HL,4AA0     ;move filespec at 4AA0H
4A03  LD DE,42F0     ;   to 42F0H
4A06  LD BC,10
4A09  LDIR

4A0B  CALL 4A80      ;output sign-on message

4A0E  LD HL,BC00     ;set floppy tape BASE to
4A11  LD (41AD),HL   ;  BC00
4A14  LD HL,BB00     ;set BASIC mem size to
4A17  LD (40B1),HL   ;  BB00

4A20  JP 4B00        ;jump to FORTH

4A80  LD A,(HL)      ;subroutine to output
4A81  OR A           ;  message pointed to
4A82  RET Z          ;  by HL.
4A83  PUSH HL        ;  0 byte terminates
4A84  CALL 33
4A87  POP HL
4A88  INC HL
4A89  JP 4A80

4AA0  DEFM 'SCREENS FTH'     ;filespec
4AAB  DEFB 0
4AAC  DEFW BF00
4AAE  DEFW 0

4AB0  DEFB 1C        ;sign-on message
4AB1  DEFB 1F
4AB2  DEFM 'Floppy Tape FORTH Version 1.1'
4ACF  DEFB 0D
4AD0  DEFB 0
```

Listing 3.

```
          0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F
-----------------------------------------------------------
4A00 -   21 A0 4A 11  F0 42 01 10  00 ED B0 CD  80 4A 21 00
4A10 -   BC 22 AD 41  21 00 BB 22  B1 40 00 00  00 00 00 00
4A20 -   C3 00 4B 00  00 00 00 00  00 00 00 00  00 00 00 00

4A80 -   7E B7 C8 E5  CD 33 00 E1  23 C3 80 4A  00 00 00 00

4AA0 -   53 43 52 45  45 4E 53 20  46 54 48 00  00 BF 00 00
4AB0 -   1C 1F 46 6C  6F 70 70 79  20 54 61 70  65 20 46 4F
4AC0 -   52 54 48 20  56 65 72 73  69 6F 6E 20  31 2E 31 00
4AD0 -   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
```

using, for example, @SAVE "SCREENS/FTH" 64, 20000. If you have Aculab's extended BASIC then use @SET "SCREENS/FTH" 63. Both these methods will create a 64-sector file, presently containing garbage, for the FORTH system to use.

2. Load your MMS cassette FORTH and editor blocks 2 to 6 of Listing 1 into any convenient blocks in your FORTH system and LOAD them. The modified FORTH system is now ready for use.

3. Assuming the floppy tape containing the "SCREENS/FTH" file is still in the drive, execute the FORTH word @LIST. This will put the floppy tape directory into its new position at the top of memory. Select floppy tape input/output using the FORTH word FLOPPY and empty the first block with 1 CLEAR. Edit the contents of block 7 of Listing 1 into block 1. Repeat this for blocks 8 and 9 of Listing 1, editing them into blocks 2 and 3. Notice that when block 3 is brought to the block buffer the UPDATEd block 1 is automatically FLUSHed to the floppy tape.

The implementation so far described requires the pre-compiled FORTH to be loaded initially from cassette and a further five FORTH blocks LOADed from cassette. You will find it much more convenient to load the modified FORTH from the floppy tape direct at one go. This is

```
BLOCK : 7

( Cassette - floppy block transfer words      1 of 2   )
: TRANSFERWORDS ;
: C>F   ( first cassette block to move, first floppy    )
        ( destination block, no of blocks to move       )
   0 DO OVER CASSETTE I + DUP BLOCK DROP
        OVER FLOPPY I +
        EDITOR COPY FLUSH @STOP
   LOOP DROP DROP ;
HEX LABEL DIDDLE 3C3F LDA 2A CPI =0 IF 20 A MVI
         ELSE 2A A MVI THEN 3C3F STA 20 E MVI RET
CODE TWBLK DE POP HL POP BC PUSH A XRA A B MOV 212 CALL
      287 CALL F1 A MVI 264 CALL 96 A MVI 264 CALL D A MOV
      264 CALL E A MOV 264 CALL
      2020 DE LXI BEGIN BEGIN M A MOV HL INX 264 CALL
      B ADD A B MOV E DCR =0 END DIDDLE CALL D DCR =0 END
      B A MOV 264 CALL 1F8 CALL BC POP NEXT      DECIMAL


BLOCK : 8

(      Cassette-floppy block transfer words  2 of 2      )

: F>C   ( first floppy block to move, first cassette    )
        ( destination block, no of blocks to move       )

      CR OVER " Position cassette to block " . TON ENTER TOFF
      CR " Set cassette to RECORD " ENTER CR FLOPPY

   0 DO OVER I + DUP . CR @BLOCK
        OVER I + TWBLK
   LOOP DROP DROP
        " Set cassette to STOP " ENTER CR   ;


BLOCK : 9

( Code for @CLEAR                                 )

: @CLEAR   ( first block no, last block no )
           FLUSH 1 + SWAP CR
           BEGIN
               ERASE-CORE DUP . CR
               DUP DUP 0 SWAP EDITOR COPY CLEAR
               FLUSH @STOP
               1 +
               OVER OVER =
           END DROP DROP
           @START ;
```

possible as follows:

1. Get the modified FORTH running as far as the end of step 2 above. Set the initial input/output mode to FLOPPY. Set the last screen used by the EDITOR to block 0 with 0 SCR!.

2. ERASE the 256-byte block of memory immediately preceding the FORTH pre-compiled core, with HEX 4A00 100 ERASE, and store the machine code for the initialisation routine given in Z80 assembly language in Listing 2. Repeated use of the FORTH word C! in the form (byte address C!) byte by byte is easy, if a little tedious. Refer to the hexadecimal dump given in Listing 3. The initialisation code is now located in the bottom 256 bytes of the second FORTH buffer.

3. Reset to BASIC and save your complete FORTH system with @SAVE"FORTH"41, 18944,18944.

The commands @CLEAR, F>C, and C>F work more quickly on floppy tapes which have been set up using the @SET command of Aculab extended BASIC, because the drive can be stopped and re-started between FORTH blocks. If you do not have this language extension it is possible to obtain a similar result by relinking the linked list in the floppy tape directory, byte by byte, and forcing the tape operating system to write the modified directory back to tape. If you feel adventurous you might try it!

Once these modifications have been made you will have many of the benefits of a disc-based FORTH.

References:

(1) Computing Today, June 1981, Page 59.
(2) Computing Today, July 1983, Page 37.
(3) Computing Today, September 1982, Page 30.

HEAVY METAL

Jamie Clary

Now this is a *big* machine. Hitachi seem to span the whole range of computing, from the ultra-tiny 3" disc drives (not used here!) to this giant of a business computer. Desk braced, we set about reviewing.

Japan may have been unable to produce a micro of (computer) world-shattering significance, but the business-end of the market is currently enjoying a somewhat predictable level of penetration with machines from companies such as Sharp and Toshiba, to name but two. Hitachi, noted white goods manufacturer and proprietors of one of the world's largest ROMing plants (Acorn do a good deal of business with them!), have found time to do all of this *AND* devise a personal computer aimed to compete directly with IBM PC/Apricot-level systems.

The Hitachi MBE 16002 Personal Computer, as the offering is known, is intended to snatch some of IBM's business, not simply because the Hitachi is PC-compatible, but because it is also said to offer a better (cheaper?) upgrade path than its competition. The Hitachi was in the office for two weeks and although nobody in the office actually rushed out and bought one, it did raise a few eyebrows and this was not simply due to its size, so read on . . .

OVERVIEW

The Hitachi MBE 16002 is a 'separates' system consisting of three units. A largish cabinet (43 cm by 30.5 cm by 23 cm for those of you concerned with footprints) houses most of the operational hardware — an Intel 8088 CPU (hardly worth mentioning it's a 16-bit machine, is it?); 128K of user RAM with parity-checking, expandable to 384K; a generous 192K of video (graphics) RAM; Centronics, RS232C, light-pen and display interfaces; twin, half-height floppy-disc drives (or half-width, since the drives are vertically mounted); and last but not least, a fan to cool it all when the going gets hot! In addition, the cabinet has a built-in paper holder which I found particularly useful, especially as there was not a lot of room left on my desk with the Hitachi sitting on it!

The keyboard is separated from the main unit by a helical cable of adequate length. The Hitachi sales blurb refers to the keyboard as being a 'detachable type' but there is no way of actually attaching it, so this is slightly exaggerated claim. The board is solidly constructed and remains firm in use — even when using the supporting legs which raise the pitch of the keyboard by approximately 10 degrees. Like the rest of the mouldings, the keyboard was a very pleasantly coloured cream and medium-brown. In all, 96 keys are present, and these are arranged in clusters — QWERTY keys, function keys, cursor control and numeric keypad all tastefully and sensibly separated. Each key, we are told, is 'ergonomically sculptured' and give a crisp and positive response.

Our system was supplied

with a colour monitor, which would add £400 to the price one would pay for the monochrome version. The display was rock-steady and the colours were faithfully represented, but the brightness control did not seem to have a great deal of effect. Display angle can be adjusted by 15 degrees in a vertical plane, by simply depressing a lever and tilting the screen to the required angle. Another nice feature of the monitor is the ability to 'sink' the brightness and contrast controls (mounted on the front) so that they lie flush with respect to the enclosure. This may seem a trivial point but at least you can set to them when you want to and *FORGET* about them just as easily, unlike most monitors which are either awkward to adjust or are easily damaged.

ROM INSIDE

Since the Hitachi is not likely to fall into the 'hobbyist' category, details such as location of the joystick ports and how reliable the cassette interface is, are omitted here because I couldn't find either of these. The only other physical component worth noting is the single 16K ROM containing the BIOS and character generator — not so much a 'clean' machine, as a slightly grubby one. The characters generated, particularly in lower case, appeared a little uneven. This was due mostly to the incongruency in body sizes of characters such as 'a' and 'b', making lines of text look pretty shabby at times.

The Hitachi, as supplied, comes complete with MS-DOS and BASIC (an implementation of GW-BASIC, I understand). MS-DOS is already well established and I didn't really have the time or the inclination to do any silly things to it. Suffice it to say that it never caused any problems.

One very curious thing I did find when I started using the system, was that for some unknown reason, the definitions of two keys on the keyboard had switched places! I found that the '@' symbol had been swapped for the ' " ' (double-quote) symbol. Consequently, accessing the '@' symbol involved pressing the ' " ' key and vice-versa — a very strange state of affairs. A telephone call to Hitachi services (UK) and all was revealed: "The keyboard you have is the European model, whereas the keyboard encoding software was originally designed for the US market".

To rectify the situation, I was told to execute a program — aptly entitled 'KEYUK' — whereafter all would be all right, which it was. This reveals two things. First, the time taken to configure the system is increased, albeit marginally. Secondly, and more importantly perhaps, this also means that the complete keyboard is software-encoded, which should allow ALL of the keys above and beyond the 10 function keys to be redefined.

PRICES

Hitachi MBE 16002 colour system RRP £2595 plus VAT
Hitachi MBE 16002 monochrome RRP £2195 plus VAT
For further information, contact Hitachi Sales (UK) Ltd, Computer Products Division, Hitachi House, Station Road, Hayes, Middlesex, UB3 4DR (telephone 01-848 8787).

BASIC

Although I find it very difficult to criticise the languages supplied with machines such as this, due mainly to the ease with which the language can be changed for another, it is unlikely that any other implementation of the same language would make full use of the machine's capabilities as that developed specifically for it, and so it is upon this basis that my critique is formed. Fortunately, the BASIC supplied with the Hitachi possessed a fairly even distribution of good and bad features, although as a result it ends up being little more than a fairly average implementation.

The first thing I discovered, and disliked about the BASIC interpreter, was that it insisted upon having spaces between reserved words and variables, making program entry unnecessarily tedious. Another interesting feature of the interpreter was that it appeared to scan the program for errors *PRIOR* to execution — something I was not used to — and I had to start expecting syntax errors to be reported before the program had even begun to execute! Hitachi BASIC was, I found, a language which demanded unexpectedly high standards of conduct from the programmer. It would not, for example, accept the following form of the FOR — NEXT loop structure:

```
10 FOR I=1 TO 10
20 IF I=7 THEN NEXT I
30 NEXT
```

This, although perfectly acceptable in most other implementations of BASIC, would not execute at all, prompting a 'NEXT without FOR' error message, suggesting that it is illegal for an iterative loop to possess more than one terminating NEXT command.

Following on from the FOR-NEXT loop structure, a nice feature of the language is that it possesses the alternative WHILE-WEND loop, although why the WHILE-DO standard was not observed is anybody's guess. No TAB(X,Y) print formatting facility is available, and the HOME command — to move the cursor to the top, left-hand side of the display — is also missing, although an identical effect can be achieved by printing ASCII code 11.

DISCS

The disc handling commands are quite adequate, MERGE and KILL being two of the more useful among them. I could not find a command to select the drive number, so an unnecessary amount of disc swapping was occurring for the duration of the review. To obtain a directory listing of all the files on a disc, the FILES command is given from BASIC. The MS-DOS equivalent command is DIR, and it is a shame that this standard has not been adopted in BASIC which, again, would have gone some way to increasing levels of consistency.

Function key definition has been made very simple to perform. KEY 1, "LIST" for example, assigns the character string LIST to function key 1. The definitions of each function key can be displayed within a window at the bottom of the display, if desired, and this function can be 'toggled' by issuing CTRL-T.

VARIABLES

Four variable types can be specified — single precision, double precision, integer and string — using the '!' (pling), '#' (hash), '%' and '$' postfixes respectively. Variables can also be swapped, using the SWAP command. This allows the valued associated with two variables to be transferred, for example SWAP X,Y. In addition, arrays can be selectively cancelled using the ERASE command — something which a good many other, smaller machines could use!

The RANDOMIZE function has a curious quirk to its nature. On encountering this instruction in a program, the machine halts, prompting for a random seed to be entered within the range + or − 32767, and this is unusual to say the least! These are all particular points — apart from these, the BASIC is more-or-less a 'standard' implementation.

GRAPHICS

The graphics capabilities of the Hitachi are quite considerable, although not necessarily in the way one would imagine. It has the ability to display a maximum of 15 colours. Why not 16? Well, the colours are arranged as light and dark variations of blue, red, green, magenta, cyan, yellow, white and . . . BLACK? We couldn't see any difference between light and dark black! All 15 colours are available with the text-only mode, but in the graphics mode, any eight colours are available from the selection of 15.

The 192K of display RAM is consumed mainly by the paging facility, which allows 32 pages of text, or four pages of 320 by 200 resolution graphics to be co-resident. The maximum graphics resolution is 640 by 200 pixels, but as a consequence, only two pages are available at this level.

The graphics commands available from BASIC include a very fast PAINT function which fills a specified area with a selected colour, and a CIRCLE command which is similarly speedy. The page switching is a very nice feature, and although

I couldn't actually get to grips with the command for switching between pages, the demonstration software showed that the facility had a lot of potential.

The display occupies approximately 80% of the total screen area, and this is used for all text and graphics output. In addition, the colour of the remaining 20% of the screen surrounding the main display window can be selected from the standard 15 colour palette. Textual displays can be selected to comply with a certain attribute — thus text can be displayed flashing, underscored etc. This is manipulated with the PALETTE function, which also defines the state of most other facets of the screen display, such as border colour and so on.

One of the nicest features of the graphics is the ability to define whether textual and graphical displays are to be treated as one, or separately, as the scrolling is performed. This allows text to scroll up the display, *OVER* the graphics without disturbing them. On the whole though, I found the graphics quite difficult and tedious to manipulate, compared with some other machines, and to achieve a simple display requires very many instructions.

SOUND

Three commands are available for generating sound — BEEP, SOUND and PLAY. BEEP gives a common-or-garden ASCII 7 (CTRL-G) type tone, while SOUND and PLAY offer much more in terms of flexibility, although the sound generation facilities are limited to a single voice. The syntax of the SOUND command is SOUND F,L where F is the frequency of the note, and L is the duration. The interesting thing is that the frequency parameter can be anything between 37 and 32768: however, values for frequency in excess of 15126 were found to be completely inaudible. This does give an unprecedented range of eight audible octaves though, which is satisfactory by any standards.

The PLAY command allows a sequence of notes to be tied together, although the specific notes and other information concerning length of time units, note-duration and octave must be expressed in string form:

```
PLAY "ABCDEFG"
```

This poses no particular problems, but it does mean that some pretty tricky string manipulation must be performed if the PLAY command is used to compose 'music' (although I use the term loosely) in real-time. Now although, as I have said, a particular octave can be specified, I found that playing a range of notes from A to G caused the octave to change at C — are we to assume from this that Hitachi R & D (Japan) sing the company anthem in the key of C?

BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8
1.7	5.4	10.8	11.0	13.0	24.2	37.1	33.9

Average: 17.13 seconds

Table 1. The results of the benchmark tests.

DOCUMENTATION

Although Hitachi have not skimped on the quantity of documentation supplied with the system — the BASIC, MS-DOS and instruction manuals are all supplied in bulky, A4 ring-binders — I think it would be fair to say that the aforementioned volumes will not win any awards for English *OR* education this year. I obviously concentrated my attention upon the BASIC manual and concluded that the explanations of BASIC command-syntax were a little too concise for my liking, and I found myself experimenting with many commands in order to discover their properties. The grammar was, at times, a little impenetrable, and references to 'flappy-discs' and 'BASIC interprinters' are cited as examples of the spelling to be expected.

CONCLUSION

The Hitachi MBE 16002 is what I am tempted to call a 'good, honest machine'. It has few surprises and in use, gives an overall impression of reliability which is, of course, of utmost importance if used as it is designed to be, in the business environment. Although I had a few reservations about the documentation — particularly the BASIC manual — it must be borne in mind that the user of a machine such as this would probably be an accomplished programmer designing software for it, or a business user who, with the exception of the instruction manual perhaps, would have little interest in the documentation.

The benchmark timings shown suggest that the machine is not quite a fast as one would expect from a 16-bit micro, but it compares favourably with its companions in the business field. On the whole, a nice machine to use. Although not market-dominating material, it should give purchasers a good RUN for their money.

This team would cost you about £45,000 p.a.

And they'd all like BUPA.

This team will cost you less than £1250 p.a.*

And it never goes sick.

Business accounting no longer need piles of paper, banks of filing, and a clerical army to control it all.

Just one small disc, a plain-English manual, and two hands.

Harness Pegasus Business Software to any of a dozen leading microcomputers, and watch.

At your command, Pegasus will record sales. Issue invoices. Update stocks. Process your payroll.

It can list, file . . . perform, in fact, any one of the thousand functions that today's complex business accounting demands.

All with perfect accuracy. And at a speed you won't believe until you see it.

Yet for all its pace, Pegasus puts you in control. It processes information only at the rate you want. Prompts you politely through every step in its programme. Not with complicated codes, but in simple, direct English.

With the versatile seven-module Pegasus system, you can start small, and grow. Begin, for example, with Sales Ledger.

Then build on it module by module, to match your needs.

It's not difficult to see why thousands of organisations large and small have purchased almost 20,000 modules in less than 2 years.

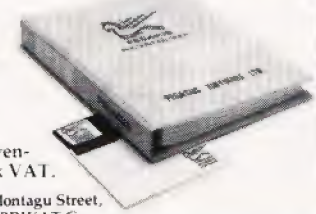
This is business software structured by experts sensitive to the user's needs, and programmed by professionals steeped in computer experience.

Any one of the 500 or so Pegasus Approved Dealers across the country is ready **now** to put Pegasus Software through its paces for you.

To start planning your team, simply clip and return the coupon.

The Seven-Module System

Sales Ledger, Purchase Ledger, Nominal Ledger, Invoicing/Sales Order Processing, Stock Control, Payroll, Job Costing.



***Based on lease purchase of a complete seven-module system over a basic 5 year term, ex VAT.**

Pegasus Software Ltd, Freepost, Brikat House, Montagu Street, Kettering. Telephone: 0536 522822 Telex: 341297 BRIKAT G

To: Pegasus Software Ltd., Freepost, Brikat House, Montagu Street, Kettering. Telephone: 0536 522822 Telex: 341297 BRIKAT G

I would like to see Pegasus Software in action. Please send me further information and tell me where I can see the system in operation.

Name ______ Title ______

Company ______

Address ______

BLOCK CAPITALS PLEASE

Tel ______ Telex ______

Type of business

Our microcomputer is ☐ Sirius ☐ Victor ☐ Apricot ☐ IBM PC ☐ DEC Rainbow ☐ NCR DMV ☐ Wang PC ☐ TI Professional ☐ Canon AS-100 ☐ CBM 8000 ☐ CBM 700 ☐ HP 150 ☐ Duet-16 ☐ We don't own a computer

No time to fill in the coupon? Just attach your business card.

PEGASUS SOFTWARE

Harness the power!

SEARCH AND REPLACE

M. C. Hart

It's unbelievably tedious having to check through a large program and change variable names. Yet it's in a large program that you're most likely to accidentally duplicate a name. Here's a routine that will make changes automatically on any Commodore machine, and can be adapted for most others using Microsoft BASIC.

If you have ever written a program and then decided that you would like to 'tidy it up a bit' by replacing all of your variable names by another variable name that is more logical, then until now you will have had to wade through all of the program to make the changes manually. Or again, you might like to make your programs absolutely 'portable' between one machine and another, and while some BASICs will accept an alphanumeric character as the second character of the variable name, some others will only accept a numeral. This actually happened to the writer on one occasion when one frequently used variable was AA$ but the machine for which it was destined would only accept A1$. Rather than altering the whole of a 31K program by hand, I developed a machine code utility to do the task for me.

The utility was actually written on a CBM 4032 PET but it has been written in such a way that it will run without alteration on practically any CBM machine(eg PETs(BASIC 2 and 4), VIC 20 and Commodore 64, but *not* the old ROM PETs without alteration). It will fit (just!) into the cassette buffer of any of these machines but is completely relocatable if you want it to reside in another part of memory or even put it into an EPROM.

HIDE AND SEEK

The logic of the program is essentially very simple, but in practice things become more complicated to implement. The program looks for a character(s) of the type specified — for example, an A — and checks to see whether the range of characters is inside either a quotes mode or a REM or a DATA statement. If it is any of the latter it carries on without making any changes until the quotes flag is turned off or the REM/DATA statement has ended,.

When it finds a match it then has to undergo a fairly extensive series of tests: for example, if we have an A which we want to replace by a Z then we would want A *and only A* to be replaced. A series of checks follows to see if the following character is an alphanumeric (eg A1), in which case we would not require a change. Similarly A would have to be distinguished from A%, A$, A(), A%(), A$(), AA, AA%, AA$,

TABLE 1

To replace variable of length 1, eg A replaced by Z
In direct mode:

```
POKE 251,ASC("A"): POKE 252,0: POKE 253,ASC("Z"):
POKE 254,0
```

(Line 2) :

```
POKE 88,0: POKE 89,0: SYS 826
```

To replace variable of length 2, eg A0;A$;A%;AA;A() by ZZ
In direct mode:

```
POKE 251,ASC("A"): POKE 252,ASC("A"): POKE
253,ASC("Z"): POKE 254,ASC("Z")
```

(Line 2) :

```
POKE 88,0: POKE 89,0: SYS 826
```

To replace variable of length 3, eg AA%;AA$;AA() by ZZ$
In direct mode:

```
POKE 251,ASC("A"): POKE 252,ASC("A"): POKE
253,ASC("Z"): POKE 254,ASC("Z")
```

(Line 2) :

```
POKE 88,ASC("$"): POKE 89,0: SYS 826
```

To replace variable of length 4, eg AA$(; AA%(by ZZ$(
In direct mode:
(Line 1) :

```
POKE 251,ASC("A"): POKE 252,ASC("A"): POKE
253,ASC("Z"): POKE 254,ASC("Z")
```

(Line 2) :

```
POKE 88,ASC("$"): POKE 89,ASC("("): SYS 826
```

General procedure

POKE first sought variable into 251	hex FB
POKE second sought variable into 252	hex FC
POKE first replacement variable into 253	hex FD
POKE second replacement variable into 254	hex FE
POKE % or $ or ('flag' into 88	hex 58
POKE ('flag' into 89	hex 59

AA%() and AA$(). The program also works in reverse, as you might expect, so that AA$ is distinguished from AA, for example.

The accompanying flowchart indicates the kind of decisions that have to be taken. It assumes a Microsoft-type BASIC line which has the following structure:

line link (low byte), line link (high byte), line number(low byte), line number (high byte), variable name . . . tokens . . . etc. (until the end of line is signified by a single zero byte)

If the type of BASIC in your machine follows this logic (if it was written by Microsoft then it almost certainly will) then you should be able to adapt this routine for your own purposes using the code and the flowchart.

INSTRUCTIONS FOR USE

The variable that you are seeking to replace I shall term 'sought variable' and the one you intend to replace it with I shall call the 'replacement variable'. To use the routine you POKE the ASCII values of the one/two 'sought' variables into locations 251 and 252 decimal and the ASCII values for the 'replacement' variables into locations 253 and 254 decimal.

To distinguish a 'simple' variable from an integer/string/array variable, use is made of other zero-page locations to act as 'flags'. If you want to replace all the AA$ (and not the AA or the AA$()) then you would poke the ASCII code for '$' into location 88 decimal. Finally, another flag is used in case you want to change *only* the dimensioned variables, for example only the AA$() and not the AA$ or AA. It is just as well to POKE these locations with zero if you do not want to use any of these flags, particularly if you are going on to change more than one variable in any one session.

The full set of instructions is given in Table 1, which ought to cover every case. A general synopsis is also given. You can POKE in the direct ASCII codes directly, of course; eg POKE 88,36 (=$); POKE 88,37 (=%), POKE 88,40 (= left hand bracket). A variable such as A$ or A% or A (ie a 'single letter' string, integer or array variable) may be treated as though it were a standard two-character variable such as AA — see example 2 in Table 1.

PROGRAM NOTES

For those interested in the 6502 machine code programming side of things, locations FB-FE are four 'free' zero page locations left unused and so available for machine code programmers in the VIC 20 and the Commodore 64 (while in the PET they are used for the pointers for the Machine Language Monitor (MLM) but may be 'borrowed' for these purposes!)

Locations 58-59 hex form part of a miscellaneous numeric work area (sometimes called FPAC#3) on both the PET and the C-64/VIC-20 normally used by the interpreter to process complex numerical calculations. In any case, they perform with no adverse effects. Finally, location 2 (part of the USER jump bytes on PETs/VIC) is left deliberately unused on a 64 and this location is used as a 'flag' to signify whether one is in quotes/REM/DATA statement or not.

CAUTION!

First, the routine as it stands will *not* handle 'long' variable names which are string or integer variables, although it will handle long variable names which are simple variables. In the interests of

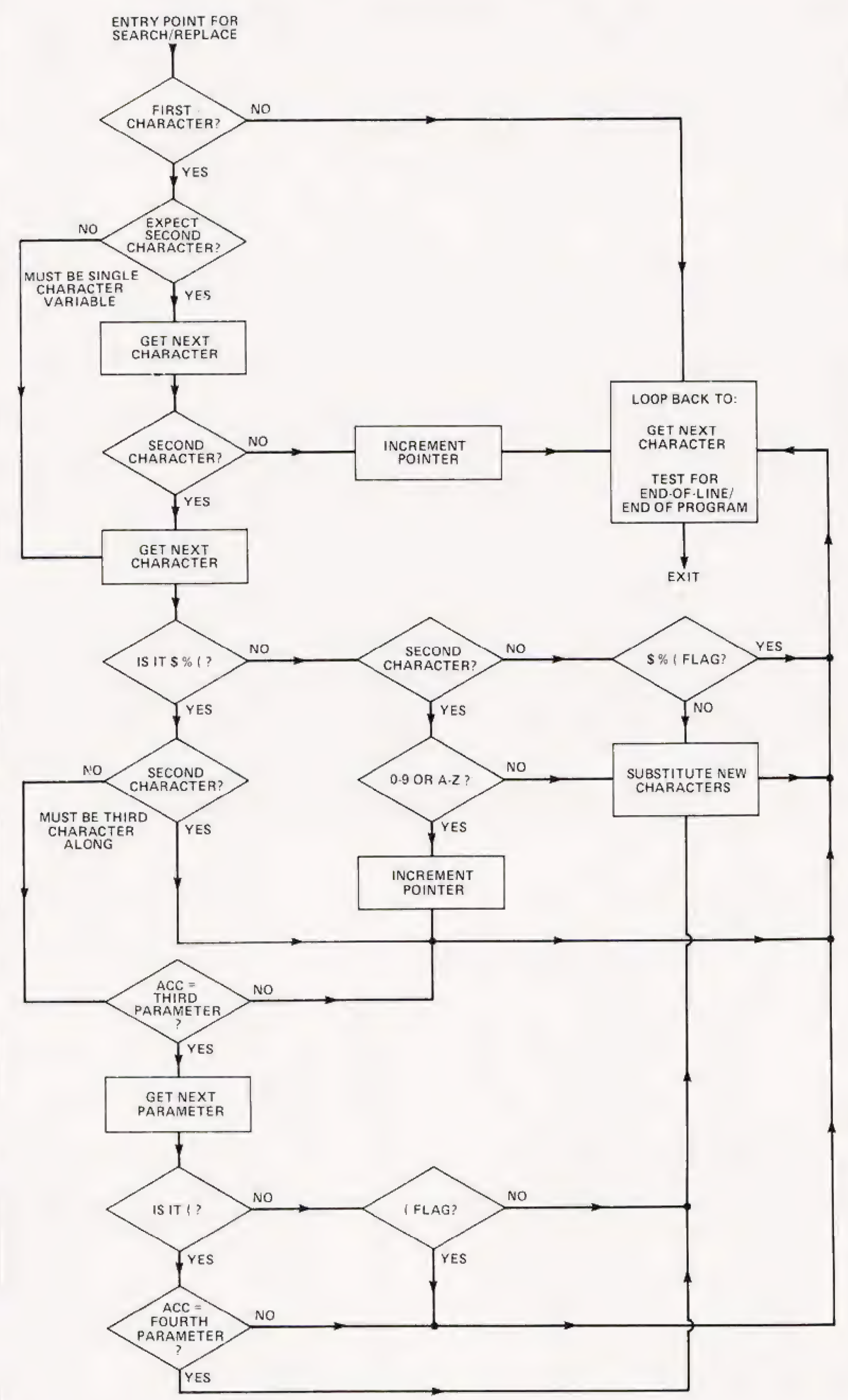


Fig. 1 Flowchart for the search-and-replace program.

Listing 1. The complete search and replace code.

```
033A A2 00    LDX #$00     ) Initialise quotes/
033C 86 02    STX $02      ) REM/DATA flag
033E A9 04    LDA #$04     ) Initialise
0340 85 24    STA $24      ) pointers
0342 A9 04    LDA #$04     ) for start
0344 85 25    STA $25      ) of BASIC text
0346 A0 01    LDY #$01     ) Set Y = 1
0348 E6 24    INC $24      ) Increment
034A D0 02    BNE $034E    ) BASIC text
034C E6 25    INC $25      ) pointers
034E A1 24    LDA ($24,X)  ) Load next byte of BASIC text
0350 F0 7D    BEQ $03CF    ) Branch if end-of-line
0352 C9 22    CMP #$22     )  Is it quotes ?
0354 F0 08    BEQ $035E    ) Yes - branch
0356 C9 83    CMP #$83     ) No - is it DATA ?
0358 F0 04    BEQ $035E    ) Yes - branch
035A C9 8F    CMP #$8F     ) No - is it REM ?
035C D0 08    BNE $0366    ) No - branch
035E 48       PHA          ) YES; quotes,DATA,REM : Save Acc.
035F A5 02    LDA $02      ) Turn quotes flag
0361 49 FF    EOR #$FF     ) into 'on' i.e. $FF
0363 85 02    STA $02      ) and store back in $02
0365 68       PLA          ) Restore Acc.
0366 E4 02    CPX $02      ) Is X zero  or $FF ?
0368 D0 DE    BNE $0348    ) If $FF (non-zero) loop back
036A C5 FB    CMP $FB      ) Is Acc. 1st sought char. ?
036C D0 DA    BNE $0348    ) No - loop back
036E A5 FC    LDA $FC      ) Yes - get next sought char.
0370 F0 0F    BEQ $0381    ) Branch if second not expected
0372 B1 24    LDA ($24),Y  ) Get next char. from text
0374 C5 FC    CMP $FC      ) Is it 2nd sought char.?
0376 F0 08    BEQ $0380    ) Yes - branch
0378 E6 24    INC $24      ) No - so increment
037A D0 02    BNE $037E    ) pointers to text and
037C E6 25    INC $25      ) loop back for another
037E D0 C6    BNE $0346    ) character
0380 C8       INY          ) Get next
0381 B1 24    LDA ($24),Y  ) character along
0383 C9 24    CMP #$24     ) Is it $ ?
0385 F0 1B    BEQ $03A2    ) Yes - branch
0387 C9 25    CMP #$25     ) No - is it  % ?
0389 F0 17    BEQ $03A2    ) Yes - branch
038B C9 28    CMP #$28     ) No -is it ( ?
038D F0 13    BEQ $03A2    ) Yes - branch
038F C0 01    CPY #$01     ) is Y = 1 (i.e. 2nd char.) ?
0391 D0 2B    BNE $03BE    ) No - branch
0393 38       SEC          ) Yes i.e. 2nd char. so
0394 E9 30    SBC #$30     ) see if 2nd character
0396 C9 0A    CMP #$0A     ) falls within the range
0398 90 DE    BCC $0378    ) 0-9 or A-Z i.e. it is
039A E9 11    SBC #$11     ) the second part of a
039C C9 1A    CMP #$1A     ) variable name. If so
039E 90 D8    BCC $0378    ) then loop to increment
03A0 B0 21    BCS $03C3    ) pointers and skip.
03A2 C0 01    CPY #$01     ) Is Y = 1 (i.e. 2nd char.) ?
03A4 F0 A0    BEQ $0346    ) Yes - branch
03A6 C5 58    CMP $58      ) Does Acc. =$;%;( flag ?
03A8 D0 9C    BNE $0346    ) No - branch
03AA C8       INY          ) Yes - get next
03AB B1 24    LDA ($24),Y  ) i.e. 4th character along
03AD 88       DEY          ) Restore Y to 1
03AE C9 28    CMP #$29     ) Is it ( ?
03B0 D0 06    BNE $03B8    ) No - branch
03B2 C5 59    CMP $59      ) Yes - does it match ( flag ?
03B4 F0 0C    BEQ $03C2    ) Yes - go off to 'replace'
03B6 D0 8E    BNE $0346    ) No - branch
03B8 A5 59    LDA $59      ) Get ( flag into Acc
03BA F0 06    BEQ $03C2    ) Branch to replace if set
03BC D0 EA    BNE $03A8    ) Loop back if unset
03BE A5 58    LDA $58      ) Get $;%;( flag into Acc.
03C0 D0 84    BNE $0346    ) Loop back to skip if unset
03C2 88       DEY          ) To restore Y to 1
03C3 A5 FD    LDA $FD      ) Load 1st replacement char.
03C5 81 24    STA ($24,X)  ) And substitute
03C7 A5 FE    LDA $FE      ) Load 2nd replacement char.
03C9 F0 D9    BEQ $03A4    ) Loop back if zero
03CB 91 24    STA ($24),Y  ) Otherwise substitute it
03CD D0 E7    BNE $03B6    ) Unconditional loop back
03CF B1 24    LDA ($24),Y  ) Acc = 0;get next along
03D1 C8       INY          ) and test for three zeros
03D2 11 24    ORA ($24),Y  ) i.e. end of program : if so,end
03D4 F0 11    BEQ $03E7    ) by branching to RTS
03D6 88       DEY          ) Restore Y to 1
03D7 18       CLC          ) Clear carry flag
03D8 86 02    STX $02      ) X=0 so put in quotes/REM/DATA flag
03DA A9 04    LDA #$04     ) Increment text pointers by
03DC 65 24    ADC $24      ) four to ensure link pointers
03DE 85 24    STA $24      ) and line numbers kept clear
03E0 8A       TXA          ) X = 0 so
03E1 65 25    ADC $25      ) process carry if any
03E3 85 25    STA $25      ) and pop back into pointer
03E5 D0 C1    BNE $03A8    ) Unconditional loop ( via stepping
03E7 60       RTS          ) Done at last!             / stone)
```

Listing 2. BASIC loader and demo program.

```
1 FOR J=826TO999:READ X:POKEJ,X:NEXT
10 A=10
20 A%=20
30 A$="30"
40 DIM A(10):DIM A%(10)
50 DIM AA(10)
60 AA=35
70 AB=40
80 AA%=50
90 AA$="60"
100 AA(1)=50
110 DIM AA$(10):DIM AA%(10)
120 :
130 REM:DATA A,A%,A$,AA,AA%,AA$,AA$(1)
140 REM  A,A%,A$,AA,AA%,AA$,AA$(1)
150 PRINT"A,A%,A$,AA,AA%,AA$,AA$(1)"
160 :
170 PRINT"A=";A
50000 DATA 162,0,134,2,169,4,133,36
50010 DATA 169,4,133,37,160,1,230,36
50020 DATA 208,2,230,37,161,36,240
50030 DATA 125,201,34,240,8,201,131
50040 DATA 240,4,201,143,208,8,72
50050 DATA 165,2,73,255,133,2,104
50060 DATA 228,2,208,222,197,251,208
50070 DATA 218,165,252,240,15,177,36
50080 DATA 197,252,240,8,230,36,208
50090 DATA 2,230,37,208,198,200,177
50100 DATA 36,201,36,240,27,201,37
50110 DATA 240,23,201,40,240,19,192
50120 DATA 1,208,43,56,233,48,201
50130 DATA 10,144,222,233,17,201,26
50140 DATA 144,216,176,33,192,1,240
50150 DATA 160,197,88,208,156,200,177
50160 DATA 36,136,201,40,208,6,197
50170 DATA 89,240,12,208,142,165,89
50180 DATA 240,6,208,234,165,88,208
50190 DATA 132,136,165,253,129,36,165
50200 DATA 254,240,217,145,36,208,231
50210 DATA 177,36,200,17,36,240,17
50220 DATA 136,24,134,2,169,4,101
50230 DATA 36,133,36,138,101,37,133
50240 DATA 37,208,193,96
```

TABLE 2

Machine	Tenth byte
VIC 20 (no expansion)	16
VIC 20 with +3K	No change
VIC 20 with +8K	18
Commodore 64	8

economy this was one refinement that had to be foregone.

Second, before you go actually changing all of your N$ to R$ for example, make sure that you have not already got an R$! The routine will change what you specify but it does not check to see if the new variable you are going to use has already been used. If you make a mistake and try to convert back again then you are going to turn your program into a mess very quickly as the 'new' and the 'old' R$ will not be distinguishable from each other.

LOADING THE CODE

As the program is 174 bytes long then the following simple loader will suffice:

```
FOR J=0 TO 173: READ X:
POKE 826+J,X: NEXT J: END
```

Obviously locations other than the cassette buffer (826 decimal) can be used. The routine is set up by default for PETs in which BASIC starts at $0401 and this routine starts scanning from just before the first byte of text, ie $0404. These bytes are the sixth and the tenth in the DATA list. For a Commodore 64 change the tenth byte into an 8, ie start at $0804. For VIC 20s, the start of BASIC is more problematic as the various memory locations alter the start of BASIC text. You can determine the start of BASIC by PEEK(43) +256*PEEK(44) in the VIC and according to published information this should be 4097 for an unexpanded VIC, 1025 for a VIC with 3K expansion and 4609 for a VIC with 8K or more expansion. However it is worth double checking these figures on your particular configuration. To adapt the DATA list for a VIC and/or C-64 refer to Table 2.

Despite the fact that the routine loads two bytes earlier than the cassette buffer in the VIC/Commodore 64 this does not create any problems (location $033A is the start of the second cassette buffer in the PETs). If a program is saved then obviously the contents of the cassette buffer in the VIC/Commodore 64 will get overwritten, but the routine can still be loaded and/or run again from memory to restore it.

As the routine is written so that it is completely relocatable, one may choose any suitable location for it and the entry point will be the first byte.

A FEAST OF FANTASY

It sometimes seems to me that life is very like an Adventure game — full of little quirks of fate and situations that seem designed to drive the poor player stark, staring mad. Fate certainly had it in for this Adventure review, which instead of being a feast, can be considered to be more of an hors d'oeuvre for the courses yet to come in future issues of Computing Today.

Our newly-absorbed magazine, MicroComputer Printout, featured regular Adventure game reviews, and it seems a good idea to continue the tradition. After all, it makes a good excuse to play the things 'in the line of duty' so to speak. We were going to have an impressive start this issue, but calamity followed disaster. The only reason we couldn't postpone this feature entirely was that it was on the front cover, and it was too late to remove it. Good old Fate!

Just a taster of the problems — would you believe that both of the BBc Micros in the office could decide that they were going to blow their cassette ports practically simultaneously? Or that the only Commodore 64 available was required for other important work? No other machine was available, even though the Commodore public relations people have their offices within five minutes walk of our offices. This was because the Libyans had, at that very moment, decided to start dispensing their own peculiar brand of diplomacy from an upstairs window of the Embassy, and said PR people were scrambling away from their office over rooftops.

So on with a rather patchy set of reviews. At least the Atari and Dragon decided to co-operate (although the Dragon exhibited a strange tendency to crash every time I tried to GO NORTH).

The Golden Baton is the first of the Mysterious Adventures from Brian Howarth, marketed by Channel 8 Software amongst others. There are 10 Adventures (so far) in the series, and versions are also available for the Commodore 64 and Dragon 32 (and 48K Spectrum, I believe). The adventures are text-only on the Atari, but lacked that depth of description available in other games. Descriptions of the locations are limited to one or two lines, which doesn't help much with the imagery.

On the other hand it is extremely fast as regards response time, flashing up the next scene almost as soon as you hit the Return key. Other types of input take a bit longer, but not drastically. I don't suppose an experienced player will have much difficulty with this game — I made it through the first dozen or so problems in less than half-an-hour. But it's quite good fun and useful to cut your teeth on if you're starting out.

On the other hand, Fishy Business from Salamander on the Dragon 32 (also available for other machines) had me a trifle stumped! I can get off the island, and I can swim underwater without drowning, but I haven't yet found any other useful locations. Just shoals of red herrings and blue kippers, that swim too fast for me to catch them.

This is a pity, because Salamander's games are characterised by little booklets with line drawings of all the locations and very intriguing they are too. (This gets round the problem of trying to pack graphics into the game, which is text-only). I know all these locations are there — I just can't get to them! Well, it'll give me something to do over Easter...

The Eye of Zoltan for the BBC and the Electron is another text-adventure, and is also very fast — and laconic — like the Golden Baton. But I feel it's a little unfair. At one point you have to guess what password gains you entry into the castle, and this could take a very long time! I tried all the likely-looking words I'd found in the game, then variations on the title, then the programmer's name: nothing worked. Eventually I lost my temper, pressed Break and listed the program (it's not protected in any way) to find the offending phrase. (I only cheat in cases of extreme distress, you understand!) Well, in a way I suppose the word is obvious, once you know it, but I wonder how many people would think of it?

Finally we come to a rather interesting situation on the Spectrum. I tried two Western games, one called Greedy Gulch from Phipps Associates and the other called Ghost Town from Virgin. They are, to put it mildly, a bit similar! Parts of the maps are the same, or very nearly so. Many of the problems are identical, and are solved in the same way using the same objects gathered in the same locations. Even the variation on how to map the desert maze is the same.

I don't know whether this is a genuine coincidence, or whether someone is being a bit naughty. I'll get in touch with the two publishers and let you know next time.

As for the games, I would recommend that, of the two, you buy the Phipps version. Ghost Town is a dreadful piece of software, riddled with appalling spelling mistakes and errors of logic. Samples: USE PUMP; YOU CAN'T. FILL BOTTLE; HOW? USE PUMP; OK. Or how about THERE IS A MATRESS (sic) WITH A PILLOW HERE. GET PILLOW; IT ISN'T HERE. Aaaarrrggh!

CP/M SOFTWARE DIRECTORY

ACCOUNTS

ACTPAK
Solicitor Systems and Services,
Systems House,
127 High Street,
Beckenham,
Kent BR3 1AG
Phone 01-658 3937 and 3938
Contact: Mr. W. Freeman
Price: £1500 (includes three days on-site training).

AUTOMATIC INVOICING
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £1000

BETPLAN
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £2000

BILLFLOW
Great Northern Computer Services Ltd,
16 Town Street,
Horsforth,
Leeds LS18 4RJ
Phone: 0532 589980
Contact: Sally Bagnal or Paul Rayner
Price: £395

BOSS-PAYPLAN
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £1000

BOSS-PAYROLL
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £500

BUSINESS DESK
Paxton Computers Ltd,
28 New Street,
St. Neots,
Huntingdon,
Cambridgeshire PE19 1AJ
Phone: 0480 213785
Telex: 32720
Contact: Jayson Brown
Price: From £595 to £2000

COSTING LEDGERS– BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

EXACT
Bickerton Computer Centre,
38a Abbey Foregate,
Shrewsbury,
Shropshire SY2 6BL
Phone: 0743 68167
Contact: Richard Bickerton
Price: £750

GENERAL LEDGER SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £350

INTERACC
Software Aids International Ltd,
16 Norval Road,
North Wembley,
Middlesex HA0 3TE
Phone: 01-904 8139
Contact: Mr. D.V. Bull
Price: From £200

INVENTORY MANAGEMENT – BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

ISL
The Byte Shop (Birmingham) Ltd
94-96 Hurst Street,
Birmingham B5 4TD
Phone: 021-622 3165
Contact: Mr. Southern
Price: £1100

NOMINAL AND PURCHASE LEDGER
Bickerton Computer Centre,
38a Abbey Foregate,
Shrewsbury,
Shropshire SY2 6BL
Phone: 0743 68167
Contact: Richard Bickerton
Price: £300

NOMINAL LEDGER
Compact Accounting Services Ltd
Cape House,
Cape PLace,
Dorking,
Surrey
Phone: 0306 887373
Contact: David Parson
Price: £325

NOMINAL LEDGER – BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

PAYROLL
Bickerton Computer Centre,
38a Abbey Foregate,
Shrewsbury,
Shropshire SY2 6BL
Phone: 0743 68167
Contact: Richard Bickerton
Price: £200

PAYROLL
Compact Accounting Services Ltd
Cape House,
Cape PLace,
Dorking,
Surrey
Phone: 0306 887373
Contact: David Parson
Price: £425

PAYROLL
Dataflow (UK) Ltd,
Unit 18,
Central Trading Estate,
Staines,
Middlesex TW18 4UX
Phone: 0784 54171
Contact: Jane Rising
Price: £500

PAYROLL – BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

PURCHASE LEDGER
Bromley Computer Consultancy,
244a High Street,
Bromley,
Kent BR1 1PQ
Phone: 01-697 8933
Telex: 896691
Contact: Mr. A. Berridge
Price: £500

PURCHASE LEDGER
Compact Accounting Services Ltd
Cape House,
Cape PLace,
Dorking,
Surrey
Phone: 0306 887373
Contact: David Parson
Price: £325

PURCHASE LEDGER
Dataflow (UK) Ltd,
Unit 18,
Central Trading Estate,
Staines,
Middlesex TW18 4UX
Phone: 0784 54171
Contact: Jane Rising
Price: £500

PURCHASE LEDGER – BUSINESS MANAGEMENT SYSYEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

PURCHASE LEDGER SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £350

SALES LEDGER – BUSINESS MANAGEMENT SYSTEMS
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

SALES LEDGER SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £350

DATABASES

BT-80
This is a Digital Research product and should be available from most software suppliers.

CARDBOX
Caxton Software Publishing Company Ltd,
10-14 Bedford Street,
Covent Garden,
London WC2E 9HE
Phone: 01-379 6502
Contact: W.D. Barrow
Price: £195

CONDOR
MOM Systems Ltd,
40-41 Windmill Street,
Gravesend,
Kent DA12 1BA
Phone: 0474 57746
Telex: 965492
Price: From £195 to £650

DATAFLEX
Equinox Computers,
16 Anning Street,
New Inn Yard,
London EC2A 3HB
Phone: 01-739 2387
Telex: 27341
Contact: Mike Kusmirak
Price: Single-user £495, multi-user £595

DATAFLOW
Dataflow (UK) Ltd,
Unit 18,
Central Trading Estate,
Staines,
Middlesex TW18 4UX
Phone: 0784 54171
Contact: Jane Rising

DATAFLOW
Great Northern Computer Services Ltd,
16 Town Street,
Horsforth,
Leeds LS18 4RJ
Phone: 0532 589980
Contact: Sally Bagnal or Paul Rayner
Price: £185

DATA MANAGEMENT SYSTEM
Compsoft Ltd,
Hallams Court,
Shamley Green,
Near Guidford,
Near Guildford,
Surrey
Phone: 0483 898545
Contact: Heather Kearsley,
Price: £195

MICROPEN
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £125

DATASTAR
This is a MicroPrp International product and should be available from most software suppliers.

DATA STORE ONE SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £100

dBASE II
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

dBASE II DATABASE MANAGEMENT SYSTEM
Sussex Microsystems Ltd,
43 East Street,
Horsham,
West Sussex
Phone: 0403 68071
Contact: Keith Gale
Price: "under £500"

FILEFORCE
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £250

GUARDIAN
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

MAILING ADDRESS
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

MAILING SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £100

MAILMASS
Vision Associates Ltd,
57 Woodham Lane,
New Haw,
Weybridge,
Surrey,
Phone: 0932 55932
Contact: Graeme Sleeman
Price:

PRISM
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

TRENDISK/1c
Microtrend UK,
PO Box 51,
Pateley Bridge,
Harrogate,
North Yorkshire HG3 5DF
Phone: 0423 711878
Contact: Ian Cummings
Price: £150

GRAPHICS

DISPLAY MANAGER
This is a Digital Research product and should be available from most software suppliers.

DATAPLOT III
Grafox Ltd,
35 St. Clements,
Oxford OX4 1AB
Phone: 0865 242597
Telex: 83147
Contact: Geordie Herbert
Price: £495

DATAPLOT+
Grafox Ltd,
35 St. Clements,
Oxford OX4 1AB
Phone: 0865 242597
Telex: 83147
Contact: Geordie Herbert
Price: £195

LANGUAGES

ALGOL-60
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

APL V80
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

BAZIC
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £124

C-BASIC
This is a Microsoft product and should be available from most software suppliers.

CBASIC COMPILER
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £333

CIS COBOL
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £425

COBOL-80
This is a Microsoft product and should be available from most software suppliers.

COMAL
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £145

POLYFORTH
Computer Solutions Ltd,
Treway House,
Hanworth Lane,
Chertsey,
Surrey KT16 9LA
Phone: 093 28 65292
Contact: Chris Stephens
Price: From £750 to £2750

FORTRAN-80
This is a Microsoft product and should be available from most software suppliers.

JRT PASCAL
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

KBASIC
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

LISP-80
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £50

M-BASIC
This is a Microsoft product and should be available from most software suppliers.

PASCAL/M
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

PASCAL/MT
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

PASCAL/MT+
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £233

PASCAL/Z
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

PRO PASCAL
Prospero Software,
37 Gwendolen Avenue,
London SW15 6EP
Phone: 01-785 6848
Contact: Tony Hetherington
Price: £220

PRO FORTRAN
Prospero Software,
37 Gwendolen Avenue,
London SW15 6EP
Phone: 01-785 6848
Contact: Tony Hetherington
Price: £220

UCSD PASCAL
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

MODELLING

CALCSTAR
This is a MicroPrp International product and should be available from most software suppliers.

FASTPLAN
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £395

PEACHCALC
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £200

PLAN80
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

SUPERCALC
Tamsys Ltd,
12a Sheet Street,
Windsor,
Berkshire SL4 1BG
Phone: 075 35 56747
Contact: David Atkinson
Price: £129

SUPERCALC I
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £126

SUPERCALC II
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £195

PROGRAMMING

ACCESS MANAGER
This is a Digital Research product and should be available from most software suppliers.

ANIMATOR
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: From £225

BASIC UTILITY DISK
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

BASKAM
Great Northern Computer Services Ltd,
16 Town Street,
Horsforth,
Leeds LS18 4RJ
Phone: 0532 589980
Contact: Sally Bagnal or Paul Rayner
Price: £95

COBOL DEVELOPMENT SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: Full system £550

CLIP
Tamsys Ltd,
12a Sheet Street,
Windsor,
Berkshire SL4 1BG
Phone: 075 35 56747
Contact: David Atkinson
Price: £95

FORMS UTILITY GENERATOR
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £120

PSORT
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

QSORT
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

SPP
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £133

T/MAKER II
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £165

TRANSAM SCIENTIFIC SUBROUTINES
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £495

TYPESTAR
Sussex Microsystems Ltd,
43 East Street,
Horsham,
West Sussex
Phone: 0403 68071
Contact: Keith Gale
Price: £95

XLT86
Xitan Systems Ltd,
23 Cumberland Place,
Southampton,
Hampshire SO1 2BB
Phone: 0703 871211
Telex: 47388
Contact: Jane Dards
Price: £100

WORD PROCESSING

BENCHMARK
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham

BRITISH SPELLGUARD
Vision Associates Ltd,
57 Woodham Lane,
New Haw,
Weybridge,
Surrey,
Phone: 0932 55932
Contact: Graeme Sleeman
Price: £179

LETTERIGHT
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

LEXICOM/2
Microtrend UK,
PO Box 51,
Pateley Bridge,
Harrogate,
North Yorkshire HG3 5DF
Phone: 0423 711878
Contact: Ian Cummings
Price: £350

MAGIC WAND
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

MAILMERGE
This is a MicroPrp International product and should be available from most software suppliers.

PEACHTEXT
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £250

SPELL
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £55

SPELLBINDER
Transam Microsystems Ltd,
59-61 Theobalds Road,
London WC1
Phone: 01-404 4554
Price: £275

SPELLING PROOFREADER
Peachtree Software International,
43-53 Moorbridge Road,
Maidenhead,
Berkshire
Phone: 0628 32711
Contact: Sales Manager
Price: £600

SPELLSTAR
This is a MicroPrp International product and should be available from most software suppliers.

TEXTWRITER III
Lifeboat Associates Ltd,
PO Box 125,
London WC2H 9LU
Phone: 01-836 9028
Telex: 893709

WORD PROCESSING SYSTEM
Interface Micro Systems,
160 High Street,
Southend-on-Sea,
Essex SS1 1JX
Phone: 0702 69961
Contact: Terry Willingham
Price: £100

WORDSTAR
This is a MicroPrp International product and should be available from most software suppliers.

NEXT MONTH

AN ARGUS SPECIALIST PUBLICATION

electronics today INTERNATIONAL

AUDIO SPECIAL TIME!

'AUDIO DESIGN' AMPLIFIER

Since the end of the series 'Audio Design', we've had a steady stream of enquiries to ask when the amplifier mentioned then will be hitting the pages. Well, the answer to all you who've asked that of us is that the first part, featuring the preamp, will appear next month. It promises to be a goody, too — your very own editor is already first in the queue for the complete kit, when this becomes available.

You would think that preamp/power amp complementary units had been around for so long that no new innovations in the basic format could be found — but you'd be wrong! Whilst we cannot claim that no one had thought of the idea before, John Linsley Hood has made a modification to the basic format that seems so obvious as to make you wonder why you didn't think of it yourself — and this is not to mention all the top-class circuitry (there will be a few surprises in the power amp circuitry in the following issue).

EPROM CARD FOR THE ORIC/ATMOS

There has been a lack of projects on these pages for the Oric (and, consequently, for the new Atmos as well), but we're just about to fix all that! This EPROM card will allow you to program EPROMs and then read and verify them, and then, if desired, to actually run the software inside them on the computer. For ease of construction, only one location on the card can be used for programming, but the card is reconfigurable, so EPROM (and the on-board RAM) can be placed as desired in the memory map, making this card a very flexible tool for firmware development.

NOVEL LOUDSPEAKER PROJECT

A new type of drive unit from a company based in Liverpool has been raising a certain amount of interest. The drive units actually have a 'lozenge-shaped flat diaphragm, driven round the edges, which should, in theory, overcome the problem of different bits of the diaphragm on a conventional speaker moving out of phase with one another — apparently the Japanese have been working on the same idea for quite some time but have yet to deliver the goods. Readers of ETI will have their chance to reach the fore-front of technology with this project.

ALL IN THE JUNE ISSUE OF ETI — ON SALE FRIDAY MAY 4th. PLACE YOUR ORDER NOW OR RISK MISSING OUT!

ARCADE

CHARTBUSTERS

BRITAIN'S SOFTWARE

CHARTS

1	ATIC ATAC (Ultimate) – Spectrum	(3)
2	JET SET WILLY (Software Projects) – Spectrum	(2)
3	SPACE PILOT (Anirog) – CBM 64	(–)
4	MANIC MINER (Software Projects) – Spectrum	(–)
5	MANIC MINER (Software Projects) – CBM 64	(–)
6	BONGO (Anirog) – VIC 20	(–)
7	LIGHT CYCLES (PSS) – Spectrum	(–)
8	FALCON PATROL (Virgin) – CBM 64	(4)
9	HUNCHBACK (Ocean) – Spectrum	(–)
10	FLYING FEATHERS (Bubble Bus) – CBM 64	(–)

NON-ARCADE

CHARTBUSTERS

BRITAIN'S SOFTWARE

CHARTS

1	TWIN KINGDOM VALLEY (Bug Byte) – CBM 64	(3)
2	GOLF 64 (Abrasco) – CBM 64	
3	HOBBIT (Melbourne House) – Spectrum	(–) / (2)
4	HOME MANAGER (Abrasco) – CBM 64	
5	GOLF SPECTRUM (Abrasco) – Spectrum	(–)
6	TWIN KINGDOM VALLEY (Bug Byte) – BBC	(–)
7	POOL (CDS) – Spectrum	(–)
8	MAD MARTHA (Microgen) – Spectrum	(–)
9	DRAGONBANE (Quicksilva) – Spectrum	(–)
10	THE VALLEY (APS Software) – VIC 20	(–)

by the ASP Market Research Group

Compiled with the assistance of Britain's leading software distributors, including: Pinnacle, SDL, PCE, Websters, PCS and Software Centre.

LUCAS MICRO

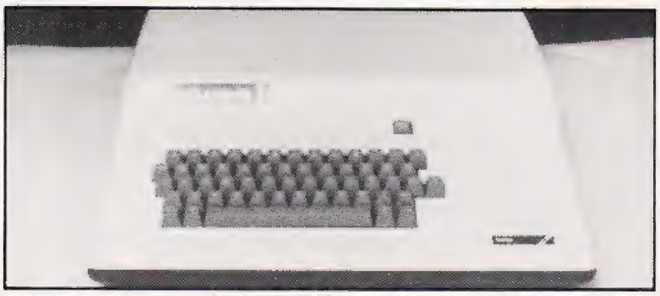
LUCAS LX

MEMORY	64K RAM expandable to 256K
LANGUAGE	Microsoft BASIC
CASSETTE	300 or 1200 baud
DISC	Single or twin 5¼ floppy disc drives DOS CP/M 2.2 (supplied) or NAS-DOS
KEYBOARD	QWERTY ☑ CURSOR ☑ NUMERIC ☑ FUNCT ☑
DISPLAY	TV ☑ MONITOR ☑ SUPPLIED ☑
INTERFACE	PARA ☑ SERIAL ☑ BUS ☑
GRAPHICS	BLOCK ☑ USER ☑
	LINE ☐ RES 392 by 256
	COLOUR 8 TEXT 80 by 25

Notes. The Lucas LX is a Z80A microcomputer aimed more at the professional and business user. Hence 5Mb Winchester disc interfacing is provided. Popular printers may be used with the RS232 serial interface, and a Centronics interface is also provided. There is an additional parallel interface connector for providing up to 16 on/off signals. The monitor supplied as standard is a 12" monochrome version: a colour monitor is also available. The high res colour graphics may be 392 by 256 in eight colours, or 784 by 256 in two colours. A wide range of applications software is available via the CP/M operating system, including Wordstar, Supercalc, and Calcstar.

NASCOM 3

MEMORY	48K RAM 14K ROM
LANGUAGE	Microsoft BASIC
CASSETTE	300 or 1200 baud
DISC	extra DOS CP/M or NAS-DOS
KEYBOARD	QWERTY ☑ CURSOR ☐ NUMERIC ☐ FUNCT ☐
DISPLAY	TV ☑ MONITOR ☑ SUPPLIED ☐
INTERFACE	PARA ☑ SERIAL ☑ BUS ☑
GRAPHICS	BLOCK ☑ USER ☑
	LINE ☐ RES 784 by 256 (two colours) 392 by 256 (four colours)
	COLOUR 8 TEXT 25 by 80
SOUND	optional



DEALER CBM MICRODEALER

DEVON

PLYMOUTH'S DEALER **nascom**
S & R BREWSTER LIMITED
86-88 Union Street, Plymouth PL1 3HG
Tel: 0752 665011 Open: 6 days

DORSET

PARKSTONE COMPUTER SYSTEMS
Main Nascom Dealer & Service Centre
Also Dealers for Gemini, Transtec, Genie
18 Station Road, Lower Parkstone, Poole, Dorset BH14 8UB.
Tel: PARKSTONE (0202) 746555

ZIPPY ELECTRONICS
West Dorsets Nascom Dealer
Business and Personal Computer Systems, Printers, Software. Tel: Bridport (0308) 56539
Mail Order Service operated. Access/Barclaycard accepted

SUFFOLK

M. D. W. ELECTRONICS
47 Woodbridge Rd. East, Ipswich IP4 5QN.
Tel: (0473) 78295
Nascom, Nasbus, Gemini, Multi-board Stockists.

WALES

BYTE-RITE
Llandaff R & TV Ltd, 24-26 High St., Llandaff, Cardiff. Tel: 0222 563760
Nascom, Gemini, Lucas Logic, C.A.D. Networks, Dragon Peripherals

WARWICKSHIRE

BUSINESS & LEISURE MICROCOMPUTERS
16 The Square, Kenilworth, Warwickshire CV8 1EB
Tel: 0926 512127
We are the largest Lucas Nascom dealer in the UK and carry the greatest variety of products for this range of computers.

CHESHIRE

Automated Business Equipment Ltd — Stockport — 061-432 0708
CBM specialist for industrial accounts, management graphics and planned maintenance.

HERTFORDSHIRE

Alpha Business Systems Ltd
Church Street, Industrial Area, Ware, Herts. Tel: 0920 68926
Specialists in recommending and supplying complete systems for small businesses.

LANCASHIRE

COMMODORE BUSINESS SYSTEMS
Commodore Home Computers Software and Hardware
01 - 228 1637
2/4 Oxford Road, Manchester M1 5QA (Opposite BBC)

LONDON

MAYFAIR MICROS
for a full range of Commodore equipment at very competitive prices. **5th Floor, 65 Duke Street, London W1. Tel: 01-629 2487**

COMPUTER CABIN
The store with everything for the enthusiast. Official Commodore Dealer and service and information centre.
24 The Parade, Silverdale, Newcastle, Staffs. Tel: 0782 636911.

TYNE & WEAR

KEY COMPUTER SERVICES LIMITED
Micro Computer solutions to everyday business problems.
Osborne House, 28 Osborne Road, Newcastle upon Tyne NE2 2AJ. Telephone: (0632) 815157
Directors: J. Sowerby, V. Shepernson.

WALES

SIGMA SYSTEMS LTD
266 North Road, CARDIFF
Tel: 621414
Also authorised dealer for CBM Digital, IBM, ACT, Sirius & Apricot

WEST MIDLANDS

MICRO BUSINESS CENTRE LTD
Wolverhampton Computer Centre, 17-19 Lichfield Street, Wolverhampton, West Midlands WV1 1EA. Tel: (0902) 29907 or 29021
Complete range of Commodore products always available.

WORCESTERSHIRE

MINSTER MICROCOMPUTERS LTD
Pegasus Integrated Accounts, Word-pro Word Processing, DMS Management Information Systems. Contact Mike Page, Sales Manager on 0562 68277. Burgale Lodge, Franche, Kidderminster, Worcs

COMMODORE 720

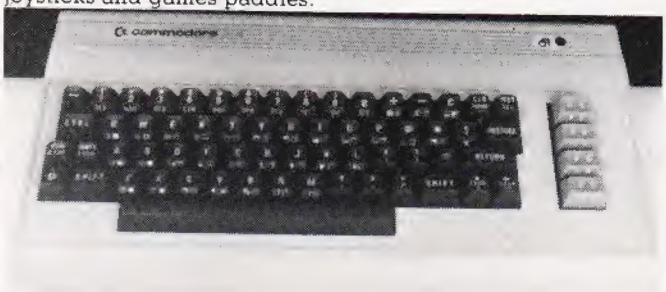
MEMORY	256K 20K ROM
LANGUAGE	Commodore BASIC
CASSETTE	300 baud
DISC	Twin in-built floppy drives
KEYBOARD	QWERTY ☑ CURSOR ☑ NUMERIC ☑ FUNCT ☑
DISPLAY	TV ☐ MONITOR SUPPLIED ☑
INTERFACE	PARA ☑ SERIAL ☑ BUS ☐
GRAPHICS	BLOCK ☑ USER ☐
	LINE ☐ RES 80 by 25
	COLOUR 16 TEXT 80 by 25
SOUND	**Three channels**

Notes. The Commodore 720 is the top model in the 700 range of business machines. It is built round the 6509 processor, but there is a dual processor (Z80 or 8088) option. The machine has been designed to meet the IEC specifications. The black-and-white monitor screen is integral and features tilt and swivel. The keyboard may be detached. The dual disc drives are built-in to the main housing and use DMA transfer, increasing speed.

COMMODORE 64

MEMORY	64K RAM 26K ROM
LANGUAGE	PET BASIC
CASSETTE	300 baud
DISC	extra DOS
KEYBOARD	QWERTY ☑ CURSOR ☑ NUMERIC ☐ FUNCT ☑
DISPLAY	TV ☑ MONITOR SUPPLIED ☐
INTERFACE	PARA ☑ SERIAL ☑ BUS ☑
GRAPHICS	BLOCK ☑ USER ☑
	LINE ☐ RES 80 by 25
	COLOUR 16 TEXT 40 by 25
SOUND	**Three channels**

Notes. The Commodore 64 is a 6510 based micro that can also use Pascal, COMAL, LOGO, FORTH and PILOT. Programs can be loaded from cassette recorder or disc drives, both extra, or cartridges. The various peripherals include printer, joysticks and games paddles.



SHARP MICRODEALER

SHARP MZ-80A

MEMORY	48K RAM	4K ROM		
LANGUAGE	Microsoft BASIC			
CASSETTE	1200 baud (built-in)			
DISC	extra	DOS		
KEYBOARD	QWERTY ☑	CURSOR ☑	NUMERIC ☑	FUNCT ☐
DISPLAY	TV ☐	MONITOR ☑	SUPPLIED ☑	
INTERFACE	PARA ☑	SERIAL ☐	BUS ☑	
GRAPHICS	BLOCK ☑	USER ☐		
	LINE ☐	RES 80 by 50		
	COLOUR	TEXT 25 by 40		
SOUND	Single channel			

Notes: The Sharp MZ-80A is a Z80 based micro. An expansion unit, printer, floppy disc unit and other peripherals are available. Other languages can also be used such as Pascal merely by replacing the tape. With the floppy disc option the machine can respond to higher level software such as Disc BASIC and FDOS (including BASIC compiler). A small range of business and educational software is available. The supplier is **Sharp Electronics (UK) Ltd.** Thorp Road, Newton Heath, Manchester M10 9BE.

SHARP MZ-80B

MEMORY	64K RAM	2K ROM		
LANGUAGE	BASIC (on tape)			
CASSETTE	1800 baud built-in			
DISC	extra	DOS		
KEYBOARD	QWERTY ☑	CURSOR ☑	NUMERIC ☑	FUNCT ☐
DISPLAY	TV ☐	MONITOR ☑	SUPPLIED ☑	
INTERFACE	PARA ☐	SERIAL ☐	BUS ☑	
GRAPHICS	BLOCK ☑	USER ☐		
	LINE ☑	RES 320 by 200		
	COLOUR	TEXT 25 by 80		
SOUND	3 channels			

Notes: The Sharp MZ-80B is a Z80A based micro. Various other languages can be loaded as the machine is "soft", no language being fitted in ROM. Expansion unit, the MZ-80P5 printer and the MZ-80FB floppy disc drive are also available. The supplier is **Sharp Electronics (UK) Ltd.** Thorp Road, Newton Heath, Manchester.

COMPUTING TODAY

Lineage: 40p per word.
Semi display: £9.00 per single column centimetre
Ring for information on series bookings/discounts.

All advertisements in this section must be prepaid
Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).

01-437 0699

Send your requirements to:
IAN ATKINSON
ASP LTD, 1 GOLDEN SQUARE,
LONDON W1

ACCESSORIES

BLANK CASSETTES!

TOP QUALITY PROFESSIONAL BRAND COMPUTER/AUDIO CASSETTES AT BUDGET PRICES

Packed in boxes of 10 cassettes complete with labels, inlay cards and library cases. Prices include VAT post & packing

Length	Box Price (10)	Qty	Amount
5 min	£4 35		
10 min	£4 90		
12 min	£4 95		
15 min	£5.00		
30 min	£5 20		
60 min	£5.80		
90 min	£7 50		

Cheque/Postal Order enclosed for £

NAME ....

ADDRESS ....

....

PROFESSIONAL MAGNETICS LTD
Dept CT, Cassette House,
329 Hunslet Rd., Leeds.
Tel: (0532) 706066

Quality Cassettes/Disks/Paper

5 x C15 Cassettes **£2.60**
10 x C15 Cassettes **£4.90**
5 x C20 Cassettes **£3.00**
10 x C20 Cassettes **£5.50**

5¼ ssdd Diskettes **£1.40** each. Listing Paper (11" x 9.5") 500 Sheets only **£5.00.** Labels, Cases & Postage Inclusive. Cheques, PO's to

C & S COMPUTER SERVICES
144 Sutcliffe Avenue, Grimsby, South Humberside DN33 1AP.

MRL PRINTERS

Chinwa CP80 195 (224 inc VAT) Star Gemini 10X (120 cps 80 col) 215 (247 inc VAT) Star Gemini 15X (120 cps 132 col) 335 (385 inc VAT) Star Delta 10 (160 cps 80 col) 359 (413 inc VAT). Juki Daisywheel (18 cps) 369 (424 inc VAT) Plus paper, ribbons & refills etc. Free delivery by Securicor

RING 0506 31605 FOR DETAILS

MRL Printer Interfaces

Available to fit most micros Electron Spectrum, Atari, Commodore 64 incl free cable, free delivery From **£39.95** inc VAT

RING 0506 31605 FOR DETAILS

Selection of Disc Drives at bargain prices from **£135** (inc VAT) to fit BBC, Tandy, Nascom etc

MICRO RESEARCH LTD (Freepost)
8 Napier Square, Houston Ind. Estate,
Livingstone West Lothian EH54 5DG

FERRANTI I.Te.C.

BBC MICRO CASSETTE LEADS

(a) 7 pin din to 7 pin din
(b) 7 pin din to 2 x 3.5 mm & 1 x 2.5 mm Jacks
(c) 7 pin din to 3 pin din & 1 x 2.5 mm Jack

ONLY £2.25 ea inc p&p/VAT

Ferranti Oldham ITeC
Department 101, Orme Mill,
Crimbles Street, Waterhead,
Oldham OL4 3JA

HARDWARE

SHARP MZ-711

64K Personal Computer

SAVE OVER **£39.00** WITH MZITOI DATA RECORDER PLUS 10 FREE GAMES
Our special price £247.00

Also Available
MZ/POI 4 colour printer **£127.**

All prices include VAT
FREE DELIVERY
Mail Order. Send Cheque/PO or quote Access no
Send SAE for software lists

HIGNETT'S Computers,
Dept PCT, 71/73 Rocky Lane,
Anfield, Liverpool L6 4BB

NEWBRAIN & SANYO

Professional Micro Computers for the price of hobby machines

Newbrain on Special Offer

SANYO 550/555 COMPUTERS

Micropro Wordstar, Calcstar etc at no extra cost! **Printers:** Epson, KDC, Juki, Shinwa, Daisystep 2000 etc. **Sanyo Monitors & Recorders.**

Call STEVENAGE (0438) 812439
anytime for hardware/software lists

Mail Order and **Access** facilities

ANGELA ENTERPRISES
4 Ninnings Lane, Rabeley Heath,
Welwyn, Herts AL6 9TD.

REPAIRS

COMMODORE REPAIRS

by CMB approved service engineers for all out of guarantee units Eg. Vic 20, CBM 64, C2N Datasette printers, Vic 20 modulators, disc units etc. For more details tel or SAE

G. C. Bunce & Son
36 Burlington Road, Burnham,
Bucks SL1 7BQ.

TEL: (06286) 6196

ADD-ONS

VIC20 CBM 64 OWNERS!!!

Buy direct from the manufact-urers and save pounds!!!

3 Slot Swichable Motherboards VIC20 **£18.95*****
4 Slot Switchable Motherboards VIC20 **£21.50*****

NEW NEW NEW

ROBOT DRIVE INTERFACE Plugs into Vics or 64S user port and gives complete computer control of DC motors. Contains all the circuitry for stop start and reverse. Comes complete with software cassette giving full instructions and control program **£21.50**

L. W.Staines & Co.,
Unit 2, Roding Trading Estate,
London Road, Barking, Essex
IG11 8BU Tel: 01-591 2900

SOFTWARE APPLICATIONS

CORTEX — FORTH

Full fig-forth with extensions for power from cortex computer. Supports Disc and cassette. 40 page manual. 16K supplied in two 2564 Eproms. Replaces 1st two Basic Eproms. **£35 inclusive**

LOMBARD SYSTEMS
18 Lombard Street, Lidlington,
Bedford MK43 0RP.

TRS-80 V. GENIE LII. Add fifteen new program commands to your Basic with our extender utility program. Machine Code £10.50. Send to: Olympic Software, 15 Arnhem Close, Aldershot, Hants GH11 1RJ

COMMODORE 64

Urgently required [illegible] applications programs [illegible] royalties or cash paid

CLARK KENT SOFTWARE,
26 North Cape Walk, Corby,
Northants NN18 9DG

COMMODORE 64 "SPRITER"

[illegible]

£3.50 [illegible]

P. STRATFORD
43 Chanctonbury Drive,
Shoreham, Sussex BN4 5FR

SERVICES

ELIMINATE FAULTY CASSETTES

DataClone is the first company in the UK established specifically for the duplication of data cassettes.

All other duplicating houses are audio orientated — only DataClone has a duplicating system designed from scratch purely to handle computer information.

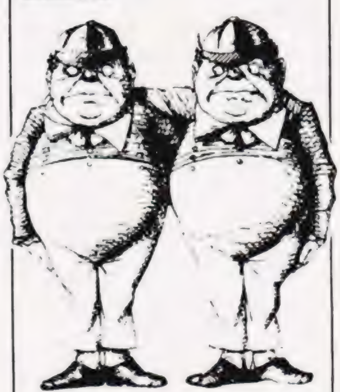
The result?

Greatly improved reliability in data transfer rates from 300 to beyond 2400 baud — previously unattainable.

All formats catered for. Quantities from 100 to infinity.

Contact us now for brochure.

DataClone — the first specialist service for computer cassettes.



DATACLONE
Unit 1, Roslin Square, Roslin Rd.,
Acton, London W3.
Tel: 01-993 2134 Telex: 21879

PROBLEMS WITH YOUR MACHINE? SCAN OUR SERVICES/ REPAIRS SECTION

COMPUTAMART

AT A GLANCE...AT A GLANCE...AT A GLANCE...AT A GLANCE...AT A GLANCE...AT A GLANCE...

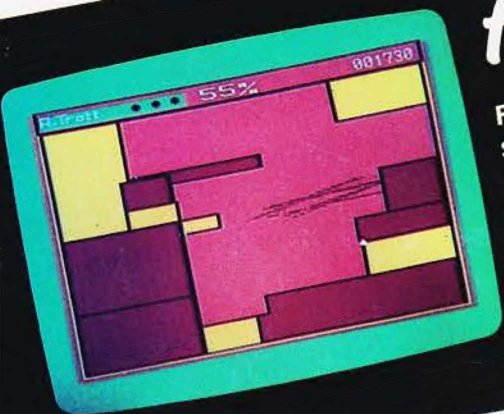
COMPUTAMART

AT A GLANCE...AT A GLANCE...AT A GLANCE...AT A GLANCE...AT A GLANCE...AT A GLANCE...

AD INDEX

SUPERSOFT

the name to remember

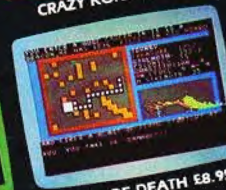


for games

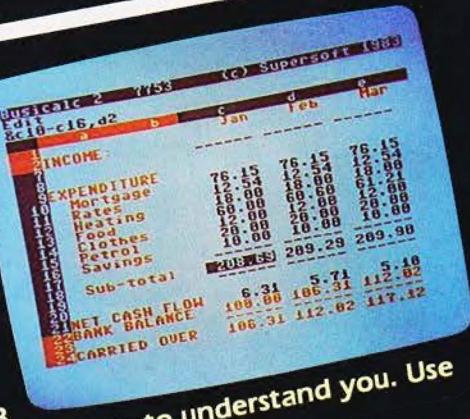
For only £8.95 you can buy a game that's exciting, soothing, and frustratingly addictive – all at the same time! STIX looks so different and sounds so different from all those other games that it will seem like being in another dimension when you sit down to play.

CRAZY KONG £8.95

WILDFIRE £6.95



HALLS OF DEATH £8.95



for business

Show your computer who's master with BUSICALC! Spreadsheet programs are used by large and small businesses to juggle with figures, prepare reports and so on. Some are very powerful indeed. The problem is that they're difficult to learn, and tricky to use – which is why we came up with the BUSICALC series.

Whether you choose BUSICALC 1, BUSICALC 2, or BUSICALC 3 you'll get a program you can understand – and one that almost seems to understand you. Use it in the home, use it for teaching, use it at work – it'll save you time and money.

for programmers

MIKRO is a full 6502/6510 ASSEMBLER with the power that professional programmers need, yet so simple to use that we recommend it to beginners! The MIKRO cartridge has many other facilities including editing commands and a machine language monitor, all for £57.50.

There's much more for the 64 in the SUPERSOFT catalogue. Ask your computer dealer for a copy, or phone 01-861 1166.

The Best 64 Software

SUPERSOFT, Winchester House, Canning Road, Wealdstone, Harrow, Middlesex, HA3 7SJ Telephone: 01-861 1166